Linear Regression

Advertising and Sales

DS 6030 | Fall 2023

regession.pdf

Contents

---

1 Introduction

Suppose that we are consultants hired by a client to provide advice on how to improve sales of a particular product. The Advertising data set consists of the product sales (in thousands of units) in 200 different markets (e.g., cities), along with advertising budgets for the product in each of those markets for three different media: TV, radio, and newspaper (in thousands of dollars).

1.1 Citation

The problem description and data are taken from **An Introduction to Statistical Learning** by James, Witten, Hastie, and Tibshirani. The advertising data is available from the textbook authors: https://www.statlearning.com/s/Advertising.csv.

2 Getting Started

2.1 Load Required Packages

```
#-- Packages
library(broom)       # for tidy model output
library(knitr)       # for kable() tables
library(GGally)      # pairs plot ggpairs()
library(tidyverse)   # dplyr, ggplot2, etc
theme_set(theme_bw()) # set default ggplot theme
library(tidymodels) # parsnip, recipes, workflow, rsample, tune, yardstick
```

2.2 Load Data

```
#: Load Data
url = 'https://www.statlearning.com/s/Advertising.csv'
advert = read_csv(url, col_select = -1) # skip first column
```

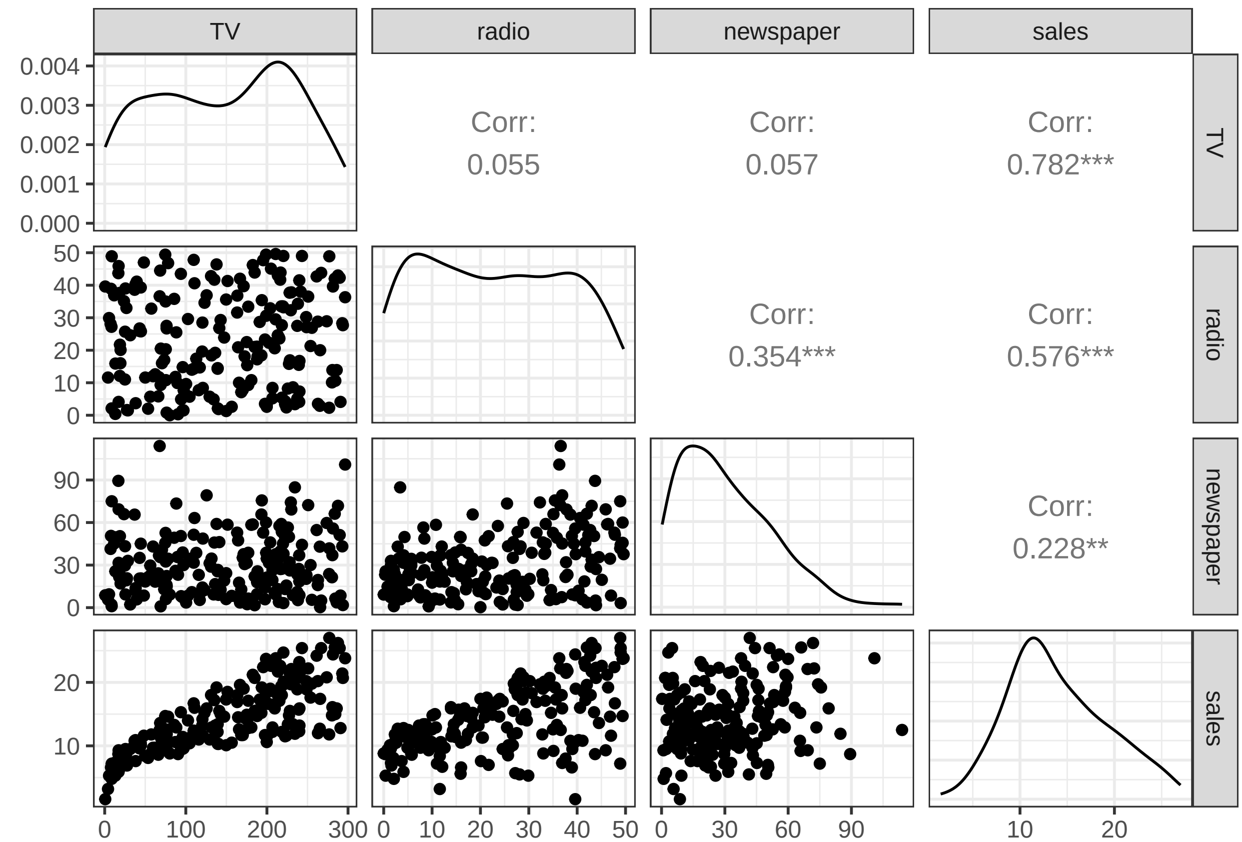
```
#: Show data
advert
#> # A tibble: 200 x 4
#>      TV radio newspaper sales
#>   <dbl> <dbl>     <dbl> <dbl>
#> 1 230.   37.8      69.2  22.1
#> 2  44.5  39.3      45.1  10.4
#> 3  17.2  45.9      69.3   9.3
#> 4 152.   41.3      58.5  18.5
#> 5 181.   10.8      58.4  12.9
#> 6   8.7  48.9      75     7.2
#> # i 194 more rows
```

3 Data Summary

```
summary(advert)
#>        TV             radio          newspaper          sales
#>  Min.   :  0.7   Min.   : 0.00   Min.   :  0.3   Min.   : 1.6
#>  1st Qu.: 74.4   1st Qu.: 9.97   1st Qu.: 12.8   1st Qu.:10.4
#>  Median :149.8   Median :22.90   Median : 25.8   Median :12.9
#>  Mean   :147.0   Mean   :23.26   Mean   : 30.6   Mean   :14.0
#>  3rd Qu.:218.8   3rd Qu.:36.52   3rd Qu.: 45.1   3rd Qu.:17.4
#>  Max.   :296.4   Max.   :49.60   Max.   :114.0   Max.   :27.0
```

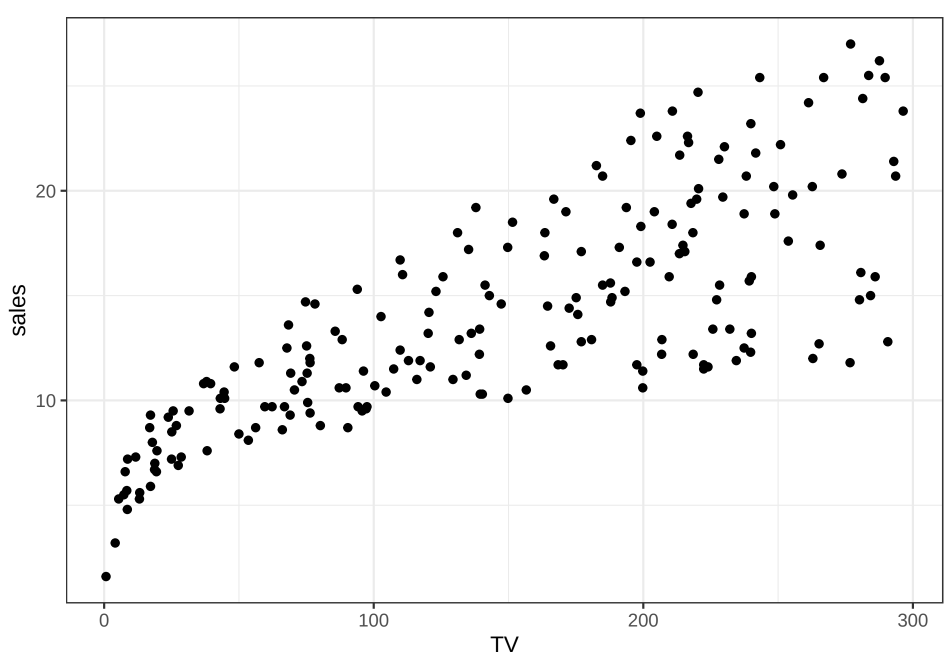
The pairs plot shows the bivariate structure

```
library(GGally)
ggpairs(advert)
```



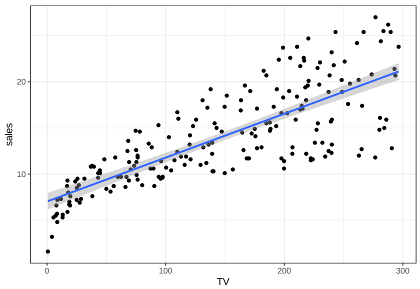
The plot of smoothed component fits help capture the relationship between the predictor variables and sales. Here is the scatterplot of TV and sales:

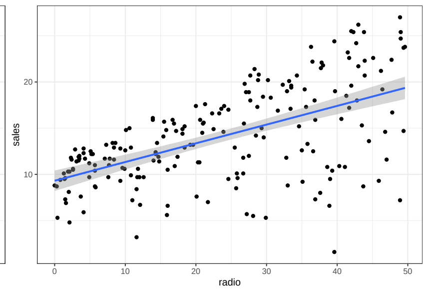
```
ggplot(advert, aes(x=TV, y=sales)) + geom_point()
```

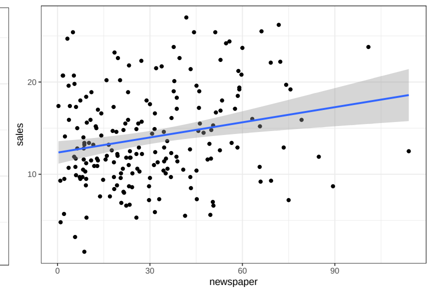


We can make three individual plots

```r
ggplot(advert, aes(TV, sales)) + geom_point() + geom_smooth(method = "lm")
ggplot(advert, aes(radio, sales)) + geom_point() + geom_smooth(method = "lm")
ggplot(advert, aes(newspaper, sales)) + geom_point() + geom_smooth(method = "lm")
```



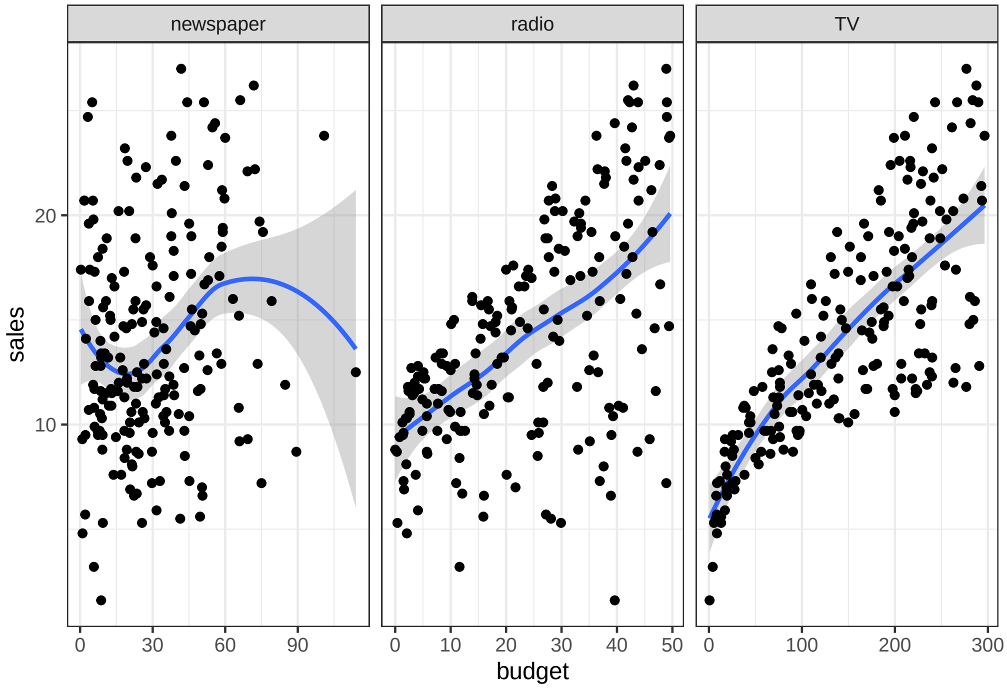




To use *faceting* to show all plots in same plot, we need to convert the data into *long* format:

```r
#: convert to "long" format
advert_long = advert %>%
  pivot_longer(
    cols = -sales,
    names_to = "predictor",
    values_to = "budget"
  )

advert_long
#> # A tibble: 600 x 3
#>   sales predictor budget
#>   <dbl> <chr>      <dbl>
#> 1  22.1 TV         230.
#> 2  22.1 radio       37.8
#> 3  22.1 newspaper   69.2
#> 4  10.4 TV          44.5
#> 5  10.4 radio       39.3
#> 6  10.4 newspaper   45.1
#> # i 594 more rows
```

```r
advert_long %>%
  ggplot(aes(x=budget, y=sales)) +
  geom_smooth() +
  geom_point() +
  facet_wrap(~predictor, scales="free_x")
```



Note: see the ggplot2 cheatsheet for help with ggplot2.

4 Regression Models

Consider the advertising sales model that uses all three predictors

$$\text{sales} = \beta_0 + \beta_1 \times (\text{TV}) + \beta_2 \times (\text{radio}) + \beta_3 \times (\text{newspaper}) + \text{error}$$

In R, the formula would be `Sales ~ TV + Radio + Newspaper` (the order of the predictor variables does not matter).

```
library(broom)   # for tidy(), glance(), and augment() functions

#- fit full (main effects) model
lm.all = lm(sales ~ TV + radio + newspaper,
            data = advert)
```

```
#- model summary
summary(lm.all)
#>
#> Call:
#> lm(formula = sales ~ TV + radio + newspaper, data = advert)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -8.828 -0.891  0.242  1.189  2.829
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  2.93889    0.31191    9.42   <2e-16 ***
#> TV           0.04576    0.00139   32.81   <2e-16 ***
#> radio        0.18853    0.00861   21.89   <2e-16 ***
#> newspaper   -0.00104    0.00587   -0.18     0.86
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.69 on 196 degrees of freedom
#> Multiple R-squared:  0.897,  Adjusted R-squared:  0.896
#> F-statistic:  570 on 3 and 196 DF,  p-value: <2e-16
```

```
#- tidy output (coefficients)
lm.all %>%
  broom::tidy(conf.int=TRUE)
#> # A tibble: 4 x 7
#>   term        estimate std.error statistic  p.value conf.low conf.high
#>   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
#> 1 (Intercept)  2.94      0.312        9.42  1.27e-17   2.32      3.55
#> 2 TV           0.0458    0.00139     32.8   1.51e-81   0.0430    0.0485
#> 3 radio        0.189     0.00861     21.9   1.51e-54   0.172     0.206
#> 4 newspaper   -0.00104   0.00587     -0.177 8.60e- 1  -0.0126    0.0105
```

```
#- tidy model summary
lm.all %>%
  broom::glance()
#> # A tibble: 1 x 12
#>   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
#>       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
#> 1     0.897         0.896  1.69      570. 1.58e-96     3  -386.  782.  799.
#> # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
#- add model output to the original data
lm.all %>% broom::augment()
#> # A tibble: 200 x 10
#>   sales    TV radio newspaper .fitted .resid   .hat .sigma  .cooksd .std.resid
#>   <dbl> <dbl> <dbl>     <dbl>   <dbl>  <dbl>  <dbl>  <dbl>    <dbl>      <dbl>
#> 1  22.1 230.   37.8      69.2    20.5  1.58  0.0252   1.69 0.00580       0.947
#> 2  10.4  44.5  39.3      45.1    12.3 -1.94  0.0194   1.68 0.00667      -1.16
#> 3   9.3  17.2  45.9      69.3    12.3 -3.01  0.0392   1.68 0.0338       -1.82
#> 4  18.5 152.   41.3      58.5    17.6  0.902 0.0166   1.69 0.00123       0.540
#> 5  12.9 181.   10.8      58.4    13.2 -0.289 0.0235   1.69 0.000181     -0.173
#> 6   7.2   8.7  48.9      75      12.5 -5.28  0.0475   1.64 0.128        -3.21
#> # i 194 more rows
```

4.1 Simpler Model

We can consider other models. You probably noticed that `newspaper` is not statistically significant in the full model, so we can try the model without it:

```
#-- Only TV + Radio
lm.TVRadio = lm(sales ~ TV + radio,
                data = advert)
summary(lm.TVRadio)
#>
#> Call:
#> lm(formula = sales ~ TV + radio, data = advert)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -8.798 -0.875  0.242  1.171  2.833
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
```

```
#> (Intercept)  2.92110    0.29449    9.92   <2e-16 ***
#> TV           0.04575    0.00139   32.91   <2e-16 ***
#> radio        0.18799    0.00804   23.38   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.68 on 197 degrees of freedom
#> Multiple R-squared:  0.897,  Adjusted R-squared:  0.896
#> F-statistic:  860 on 2 and 197 DF,  p-value: <2e-16
```

4.2 Interaction Effects

We have found that the best model so far is the one that uses `TV` and `Radio` to predict the value of `Sales`.

Specifically, the least squares model is:

$$\widehat{\text{sales}} = 2.921 + 0.046 \times (\text{TV}) + 0.188 \times (\text{radio})$$

- So a one unit increase in `TV` would suggest a 0.046 unit increase in `Sales`, no matter the budget allocated to `Radio`
- But what if spending money on `Radio` advertising actually increases the effectiveness of the `TV` advertising?
  - So `TV` effects should increase as `Radio` increases
  - E.g., spending 1/2 of a $100,000 budget on `TV` and `Radio` may increase `Sales` more than allocating the entire amount to only `TV` or only `Radio`
  - In marketing, this is the *synergy* effect. In statistics, this is known as an *interaction* effect.

4.2.1 Modeling Interactions

Consider the linear regression model with two variables and an interaction effect

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$

This model relaxes the linearity (multiply X_1 and X_2) , while maintaining the interpretable additive structure. Consider the equation re-written

$$\begin{aligned} Y &= \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon \\ &= \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon \end{aligned}$$

where $\tilde{\beta}_1 = (\beta_1 + \beta_3 X_2)$.

- Since $\tilde{\beta}_1$ changes with X_2, the effect of X_1 on Y is no longer constant.
  - Adjusting X_2 will change the impact of X_1 on Y

In R, use the notation `X_1:X_2` to include an interaction effect:

```
lm.synergy = lm(sales ~ TV + radio + TV:radio,
                data=advert)
summary(lm.synergy)
#>
#> Call:
```

```
#> lm(formula = sales ~ TV + radio + TV:radio, data = advert)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -6.337 -0.403  0.183  0.595  1.525
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 6.75e+00   2.48e-01   27.23   <2e-16 ***
#> TV          1.91e-02   1.50e-03   12.70   <2e-16 ***
#> radio       2.89e-02   8.91e-03    3.24   0.0014 **
#> TV:radio    1.09e-03   5.24e-05   20.73   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.944 on 196 degrees of freedom
#> Multiple R-squared:  0.968,  Adjusted R-squared:  0.967
#> F-statistic: 1.96e+03 on 3 and 196 DF,  p-value: <2e-16
```

4.3 Prediction

In R, the `predict()` function will return the predicted values from a fitted regression model. Besides the model, the function needs the X values (the `newdata` argument) for making predictions.
- Type `?predict.lm` to read the help pages
- The `object` argument is the `lm` model
- The `newdata` must be a `data.frame/tibble`

```
#-- predict the Sales for a budget with TV = 50
#   ($50,000) and Radio = 20 ($20,000)
pred_data = tibble(TV=50, radio=20, newspaper=0)
predict(
  lm.TVRadio,  # fitted model
  pred_data,   # pass the data in as second argument
  # other specifications, like returning prediction or confidence intervals
  interval = "prediction"  # or "confidence"
  )
#>     fit   lwr  upr
#> 1 8.969 5.634 12.3
```

Multiple values (grid of predictor values)

```
#- grid of values at which to predict
pred_grid =
  expand_grid(
    TV = seq(0, 50, by=25),
    radio = seq(0, 60, by=20),
    newspaper = 0
  )

#- add predictions to the grid (using bind_cols)
bind_cols(
  pred_grid,
  predict(lm.TVRadio, pred_grid, interval = "prediction") %>% as_tibble()
)
#> # A tibble: 12 x 6
#>      TV radio newspaper   fit    lwr   upr
#>   <dbl> <dbl>     <dbl> <dbl>  <dbl> <dbl>
#> 1     0     0         0  2.92 -0.445  6.29
```

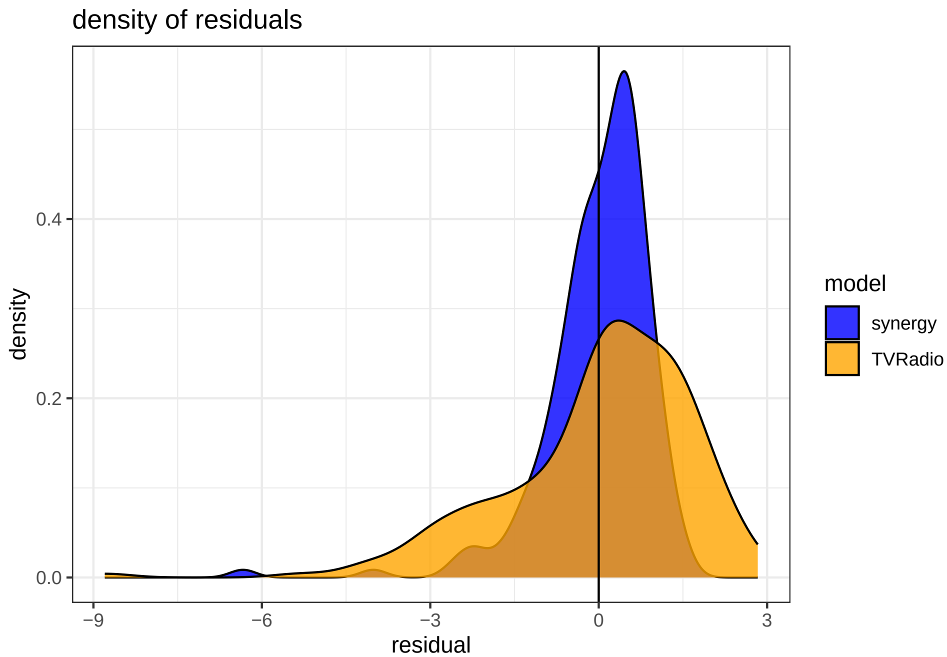
```
#> 2      0     20          0   6.68   3.33   10.0
#> 3      0     40          0  10.4    7.08   13.8
#> 4      0     60          0  14.2   10.8    17.6
#> 5     25      0          0   4.06   0.706   7.42
#> 6     25     20          0   7.82   4.48   11.2
#> # i 6 more rows

#- add predictions to the grid (using augment)
augment(lm.TVRadio, newdata = pred_grid, interval = "prediction")
#> # A tibble: 12 x 6
#>       TV radio newspaper .fitted .lower .upper
#>    <dbl> <dbl>     <dbl>   <dbl>  <dbl>  <dbl>
#> 1      0     0         0    2.92 -0.445   6.29
#> 2      0    20         0    6.68  3.33   10.0
#> 3      0    40         0   10.4   7.08   13.8
#> 4      0    60         0   14.2  10.8    17.6
#> 5     25     0         0    4.06  0.706   7.42
#> 6     25    20         0    7.82  4.48   11.2
#> # i 6 more rows

#- add predictions to the grid (using mutate)
pred_grid %>%
  mutate(
    TVRadio = predict(lm.TVRadio, .),
    synergy = predict(lm.synergy, .)
  )
#> # A tibble: 12 x 5
#>       TV radio newspaper TVRadio synergy
#>    <dbl> <dbl>     <dbl>   <dbl>   <dbl>
#> 1      0     0         0    2.92    6.75
#> 2      0    20         0    6.68    7.33
#> 3      0    40         0   10.4     7.90
#> 4      0    60         0   14.2     8.48
#> 5     25     0         0    4.06    7.23
#> 6     25    20         0    7.82    8.35
#> # i 6 more rows
```

4.4 Residual Analysis

```
# Residual Analysis
residuals_wide = advert %>%
  mutate(
    TVRadio = lm.TVRadio$residuals,
    synergy = lm.synergy$residuals
  )

residuals = residuals_wide %>%
  pivot_longer(c(TVRadio, synergy), names_to = "model", values_to = "r")
```

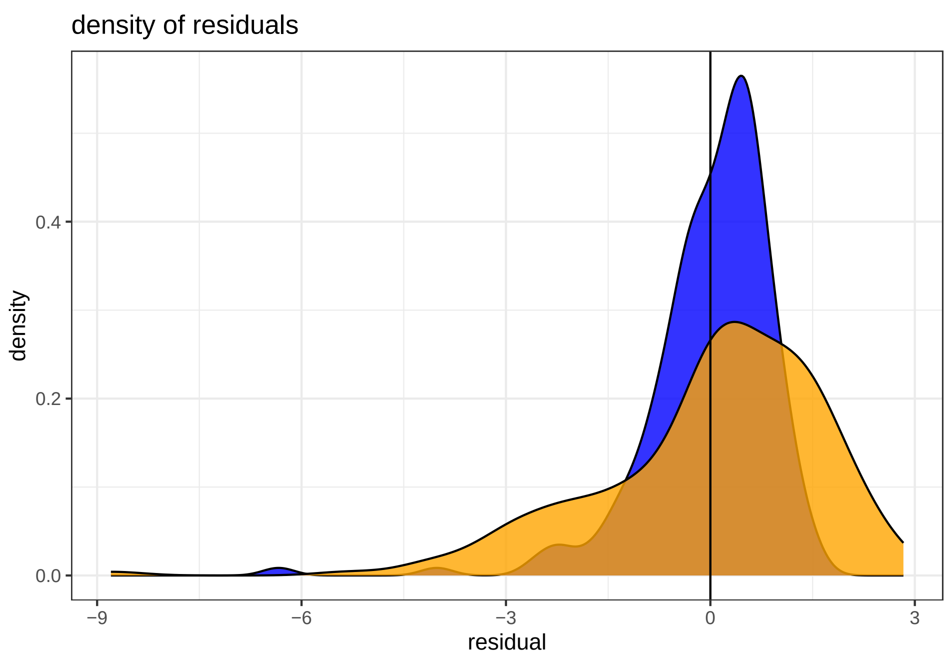
The residuals are not centered around 0 and have a long left tail, even when using the interaction model.

```
#-- Density of residuals
ggplot(residuals, aes(r, fill=model)) + geom_density(alpha=.8) +
  geom_vline(xintercept=0) +
  scale_fill_manual(values=c("blue", "orange")) +
  labs(x="residual", title="density of residuals")
```



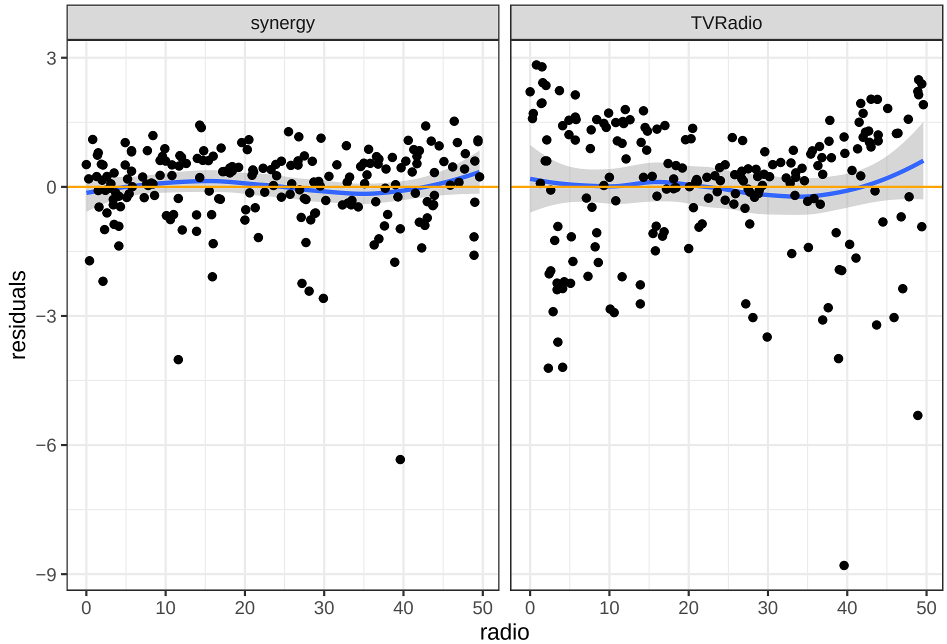
NOTE: I could still use the "wide" data and two geoms:

```
residuals_wide %>%
  ggplot() +
  geom_density(aes(synergy), fill = "blue", alpha=.8) +
  geom_density(aes(TVRadio), fill = "orange", alpha=.8) +
  geom_vline(xintercept=0) +
  labs(x="residual", title="density of residuals")
```



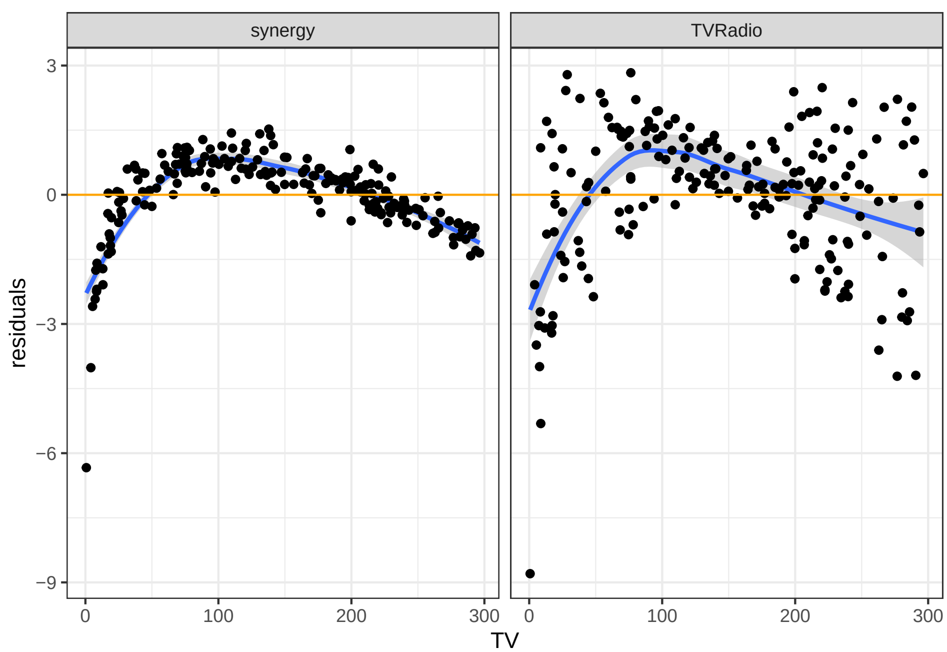
The residuals on `radio` looks decent (especially on synergy model)

```
#-- Radio: residual scatterplot
ggplot(residuals, aes(radio, r)) + geom_smooth() + geom_point() +
    geom_hline(yintercept=0, color="orange") +
    facet_wrap(~model) +  labs(y="residuals")
```



But the residuals on TV shows some unaccounted for patterns, even with synergy model.

```
#-- TV: residual scatterplot
ggplot(residuals, aes(TV, r)) + geom_smooth() + geom_point() +
  geom_hline(yintercept=0, color="orange") +
  facet_wrap(~model) +  labs(y="residuals")
```



This suggests add a transformation to `TV`, perhaps a log transformation

```
#-- Best models of ones I considered

#-- use log(TV) only
```

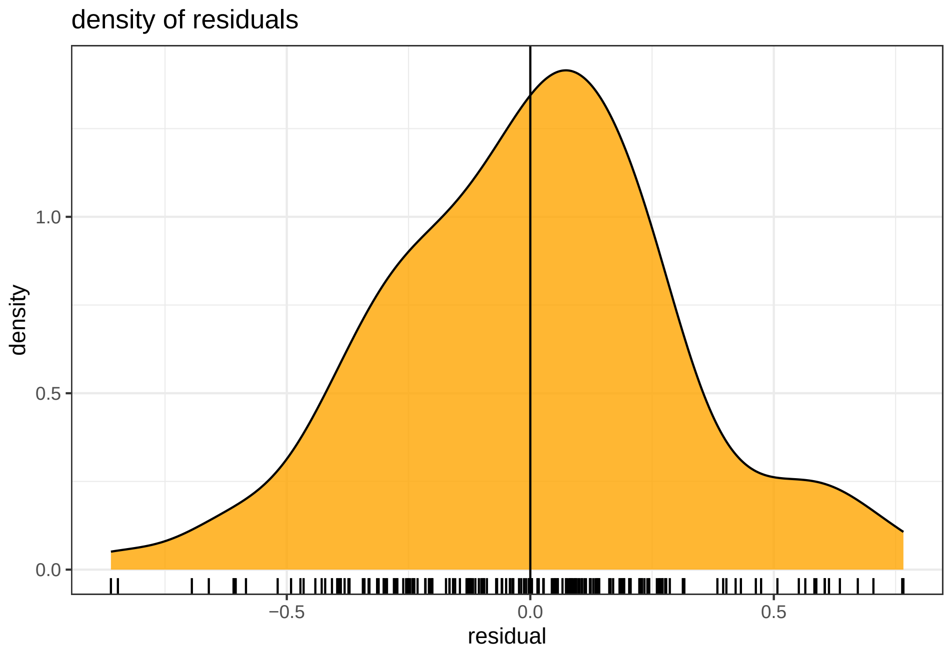
```r
best1 = lm(sales ~ log(TV) + radio + TV:radio, data=advert)
summary(best1)
#>
#> Call:
#> lm(formula = sales ~ log(TV) + radio + TV:radio, data = advert)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -0.861 -0.207  0.009  0.191  0.766
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 1.89e-01   1.68e-01    1.13     0.26
#> log(TV)     1.97e+00   3.45e-02   57.04   <2e-16 ***
#> radio       4.58e-02   2.63e-03   17.41   <2e-16 ***
#> radio:TV    1.03e-03   1.41e-05   72.76   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.304 on 196 degrees of freedom
#> Multiple R-squared:  0.997,  Adjusted R-squared:  0.997
#> F-statistic: 1.95e+04 on 3 and 196 DF,  p-value: <2e-16

#-- include main effects log(TV) + TV
best2 = lm(sales ~ TV + log(TV) + radio + TV:radio, data=advert)
summary(best2)
#>
#> Call:
#> lm(formula = sales ~ TV + log(TV) + radio + TV:radio, data = advert)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -0.8561 -0.2055  0.0161  0.1901  0.7739
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 0.231045   0.177423    1.30     0.19
#> TV          0.000489   0.000663    0.74     0.46
#> log(TV)     1.943232   0.047225   41.15   <2e-16 ***
#> radio       0.046680   0.002902   16.09   <2e-16 ***
#> TV:radio    0.001019   0.000017   60.02   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.304 on 195 degrees of freedom
#> Multiple R-squared:  0.997,  Adjusted R-squared:  0.997
#> F-statistic: 1.46e+04 on 4 and 195 DF,  p-value: <2e-16
```

With an R^2 of 0.997, we may have found how the data were generated!

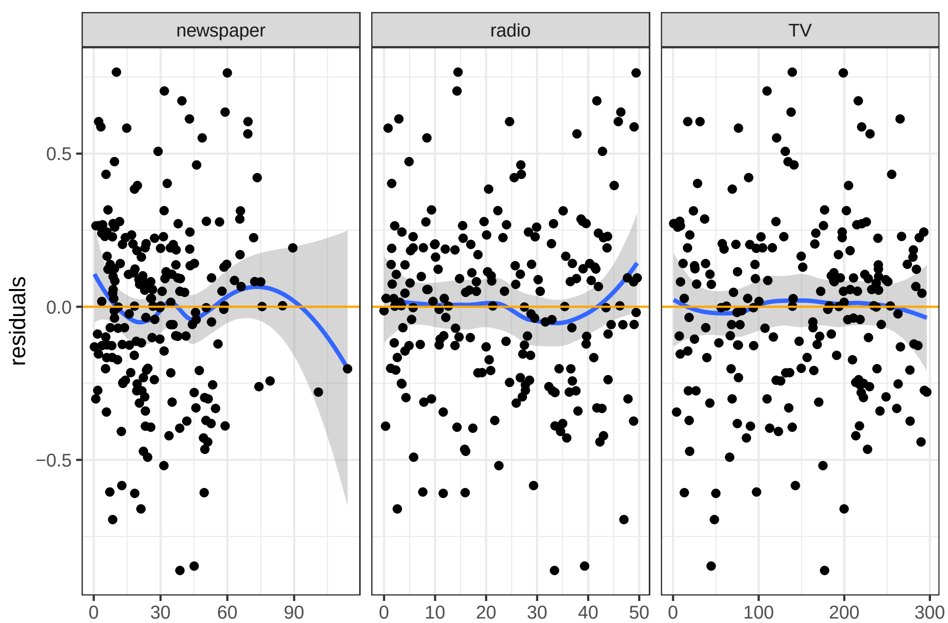
```r
R = mutate(advert, r=best1$residuals)
```

The residuals using the best model look much cleaner

```r
#-- Density of residuals
ggplot(R, aes(r)) + geom_density(alpha=.8, fill="orange") +
  geom_rug() +
  geom_vline(xintercept=0) +
  labs(x="residual", title="density of residuals")
```




```
#-- residual scatterplot
R %>%
  pivot_longer(c(TV, radio, newspaper),
               names_to = "variable", values_to="budget") %>%
  ggplot(aes(budget, r)) + geom_smooth() + geom_point() +
    geom_hline(yintercept=0, color="orange") +
    facet_wrap(~variable, scales="free_x") +  labs(x="", y="residuals")
```



5 Appendix: using `tidymodels`

The set of `tidymodels` packages https://www.tidymodels.org/ attempts to standardize R's statistical modeling and machine learning. I'll introduce the tidymodels process to replicate what we did above.

The first thing to do is to *specify* the linear model using the `parsnip::linear_reg()` function from the `parsnip` package.

```
library(tidymodels) # parsnip, broom, recipes, etc.
linear_model = linear_reg()
```

It may seem strange to specify just a plain model, but will eventually allow us to easily fit more advanced model (e.g., penalized linear regression).

Now we need to *fit* (i.e., estimate the model parameters) all our candidate models. The `parsnip::fit()` function carries out the estimation:

```
lm.all = fit(linear_model,
             formula = sales ~ TV + radio + newspaper,
             data = advert)

lm.TVRadio = fit(linear_model,
                 sales ~ TV + radio,
                 data = advert)

lm.synergy = fit(linear_model,
                 sales ~ TV + radio + TV:radio,
                  data = advert)

best1 = fit(linear_model,
            sales ~ log(TV) + radio + TV:radio,
            data = advert)

best2 = fit(linear_model,
            sales ~ TV + log(TV) + radio + TV:radio,
            data = advert)
```

The `broom` package functions (`tidy()`, `glance()`, `augment()`) still work as expected:

```
best1 %>% tidy()
#> # A tibble: 4 x 5
#>   term        estimate std.error statistic   p.value
#>   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
#> 1 (Intercept)  0.189   0.168          1.13 2.62e-  1
#> 2 log(TV)      1.97    0.0345        57.0  5.09e-124
#> 3 radio        0.0458  0.00263       17.4  1.20e- 41
#> 4 radio:TV     0.00103 0.0000141     72.8  8.52e-144
```

```
bind_rows(
  best1 = best1 %>% glance(),
  best2 = best2 %>% glance(),
  .id = "model"
)
#> # A tibble: 2 x 13
#>   model r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC
#>   <chr>     <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl> <dbl>
#> 1 best1     0.997         0.997 0.304    19521. 2.10e-242     3  -43.4  96.8
#> 2 best2     0.997         0.997 0.304    14607. 2.43e-240     4  -43.1  98.2
#> # i 4 more variables: BIC <dbl>, deviance <dbl>, df.residual <int>, nobs <int>
```

```
best1 %>% augment(pred_grid)
#> # A tibble: 12 x 4
#>      TV radio newspaper  .pred
#>   <dbl> <dbl>     <dbl>  <dbl>
#> 1     0     0         0 -Inf
#> 2     0    20         0 -Inf
```

```
#> 3      0     40          0 -Inf
#> 4      0     60          0 -Inf
#> 5     25      0          0    6.52
#> 6     25     20          0    7.95
#> # i 6 more rows
```

And the `parsnip::predict()` (`?predict.model_fit`) is very similar to the regular predict function. It requires the arguments:

- `object` is the fitted model
- `new_data` is the new data to make predictions on. Note the underscore!
- `type` depends on the model. For regression it can be `numeric` (default), `conf_int`, `pred_int`, or `raw`.

```
predict(best1, new_data = head(advert))
#> # A tibble: 6 x 1
#>   .pred
#>   <dbl>
#> 1 21.5
#> 2 11.2
#> 3  8.70
#> 4 18.4
#> 5 12.9
#> 6  7.12
```

```
predict(best1, new_data = head(advert), type = "conf_int") # confidence interval
#> # A tibble: 6 x 2
#>   .pred_lower .pred_upper
#>         <dbl>       <dbl>
#> 1       21.5        21.6
#> 2       11.1        11.3
#> 3        8.56        8.83
#> 4       18.3        18.4
#> 5       12.8        13.0
#> 6        6.97        7.27
predict(best1, new_data = head(advert), type = "pred_int") # prediction interval
#> # A tibble: 6 x 2
#>   .pred_lower .pred_upper
#>         <dbl>       <dbl>
#> 1       20.9        22.1
#> 2       10.6        11.9
#> 3        8.08        9.31
#> 4       17.8        19.0
#> 5       12.3        13.5
#> 6        6.50        7.74
predict(best1, new_data = head(advert), type = "raw")      # vector predictions
#>      1      2      3      4      5      6
#> 21.535 11.247  8.695 18.371 12.908  7.119
```