Design and Analysis of Experiments with R

A Modern Approach

D. L. Burrell

2021-07-08

Contents

Chapter 1

Prerequisites

Chapter 2

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

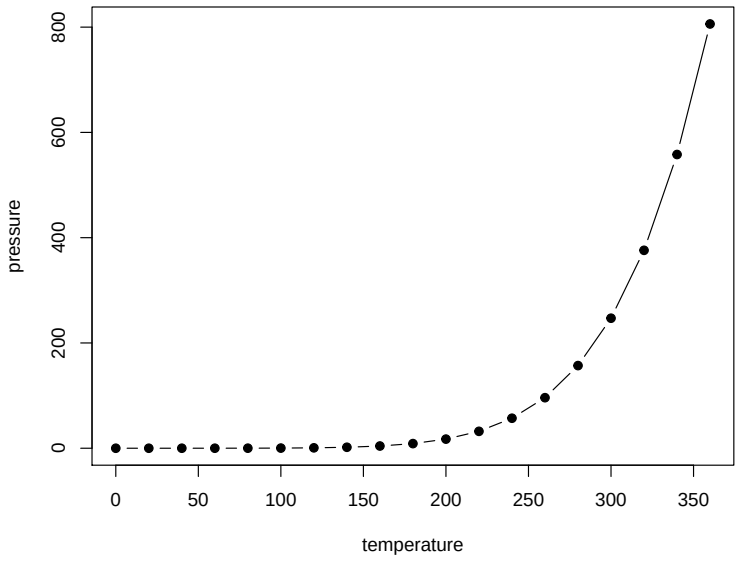


Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2021) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 3

Literature

Here is a review of existing methods.

Chapter 4

Statistical Models

One of the great scientific achievements humanity has made is the ability to conceptualize complex phenomena and processes in the natural world through models that amplify the pertinent facts and relationships of interest and provide a logical platform from which to study systems that drive decision-making. Most scientific models can be expressed as mathematical abstractions. Two broad classes of mathematical models are **deterministic models** and **stochastic models**. Broadly speaking, deterministic models leave nothing to chance and blatantly ignore uncertainties. For example the concentration c of a pollutant in a river at a point x and time t can be modeled as:

$$c(x,t) = c_0(x - vt)e^{-kt},$$

where $c_0(x)$ is the initial pollutant concentration at point x, v is the water velocity, and k is a proportionality constant, measuring the efficiency of bacterial decomposition of the pollutant. This is a deterministic model: given inputs x, v and c_0, the pollutant concentration at location x and time t is predicted with certainty. Such a model makes a very strong assumption that the model is correct and the inputs are measured with perfect precision and accuracy. But the pollutant concentration makes certain assumptions: (i) that the pollutant concentration is uniform in all directions except downstream flow, (ii) there are no diffusive effects due to to contour irregularities and turbulence, (iii) the pollutant decays as a negative exponential due to bacterial action, (iv) the bacterial efficiency is time-homogeneous, and (v) there is thorough mixing of the pollutant in the water. These assumptions are reasonable, but they not necessarily (always) true. The uncertainty of the effects at a particular location along the river and point in time can be incorporated by casting the model stochastically:

$$c(x,t) = c_0(x - vt)e^{-kt} + e$$

where $e \sim D(0, \sigma^2)$, for some probability distribution D. Allowing for the random deviation e we are now claiming that $c(x,t)$ is a random variable with

expectation:

$$\mathbb{E}[c(x,t)] = c_0(x - vt)e^{-kt}$$

4.1 Linear Models in R

Most R functions uses a formula representation to represent regression models (for continuous variables) or means and effects models (for categorical variables):

```
response ~ model
```

where the tilde (~) defines a model formula and `model` represents a set of terms to include as predictors in the model. Terms are included by their variable names and various operators such as `+` (include in model), `-` (exclude from model), `:` (interaction), `*` (full factorial structure), and so on. The intercept term (which can be explicitly denoted in a model as a `1`) is implicitly defined and need not be specified (although I often do for completeness). Thus the following model formulae are equivalent (all include the intercept):

```
Y ~ X
Y ~ 1 + X
Y ~ X + 1
```

whereas these formulae express different ways of excluding the intercept:

```
Y ~ -1 + X
Y ~ X - 1
```

Linear models are fitted using the method of ordinary least squares by providing the model formula as an argument to the `lm()` function. For example, we could generate a fictitious response variable (`Y`) and a fictitious continuous predictor (`X`) as follows:

```
# Load packages
if(!require(pacman)) install.packages("pacman")
```

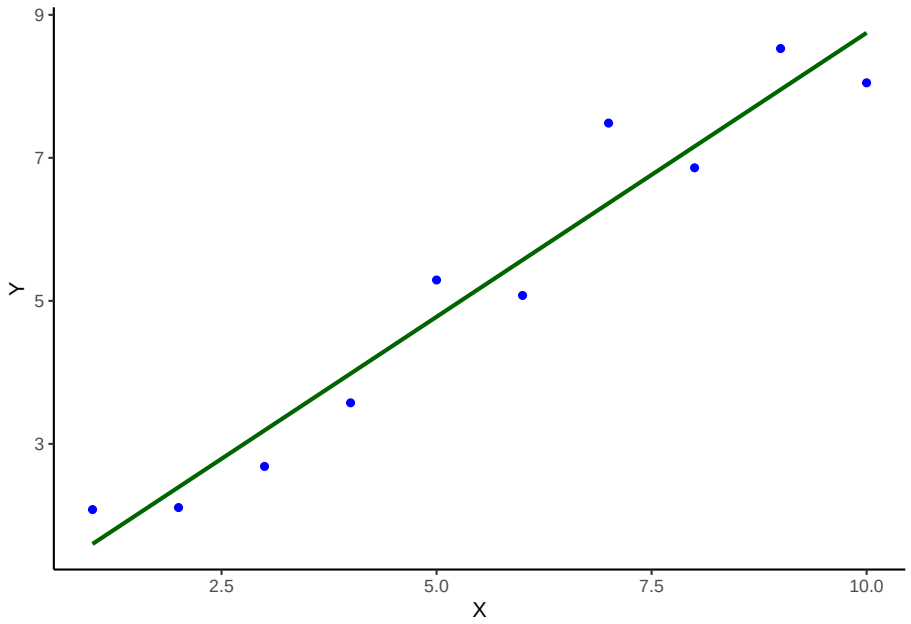
```
## Loading required package: pacman
```

```
library(pacman)
p_load(tidyverse)

# Generate data set
N = 10
fictitious_data = tibble(
   X = 1:N,
   Y = 0.9*X + rnorm(n=N, mean=0, sd=0.5)
)

# Plot data
ggplot(data = fictitious_data,
```

```
      mapping = aes(x=X, y=Y)) +
   geom_point(colour = "blue") +
   geom_smooth(method = "lm",
               formula = y ~ x,
               se = FALSE,
               colour = "darkgreen") +
   theme_classic()
```



The plot shows the data points and a simple linear regression line fit through them. To fit the regression in such a way as to be able to work with the estimated parameters we use `lm()`:

```
# Fit simple linear regression model
fictitious_lm = lm(
   formula = Y ~ 1 + X,
   data = fictitious_data)
```

To examine the estimated parameters (and hypothesis tests) from the fitted model we can use the function `summary()`:

```
# Summary of regression
summary(fictitious_lm)
```

```
##
## Call:
## lm(formula = Y ~ 1 + X, data = fictitious_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6988 -0.4738 -0.2918  0.5061  1.1202
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.80616    0.44577   1.808    0.108
## X            0.79420    0.07184  11.055    4e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6525 on 8 degrees of freedom
## Multiple R-squared:  0.9386, Adjusted R-squared:  0.9309
## F-statistic: 122.2 on 1 and 8 DF,  p-value: 3.996e-06
```

There is a lot of useful information presented here, however it is not necessarily easy to access and work with in successive computations. This is where the `broom` package comes in. There are three main functions that allow us to extract pertinent information and store it in data frames, for easy access later. The functions are:

- `tidy()`: returns the statistical findings of the model (such as coefficients)
- `glance()`: returns a concise one-row summary of the model
- ‘`augment()`: adds prediction columns to the data being modeled

Using `tidy(fictitious_lm)` we can extract the pertitent statistical information such as estimates, standard errors, test statistics and p-values:

```
# Ensure broom package is loaded
p_load(broom)

# Set output digits to 3 for printing
options(digits=3)

# tidy() output
tidy(x = fictitious_lm,
     conf.int = TRUE,
     conf.level = 0.95)
```

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic    p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>      <dbl>    <dbl>     <dbl>
## 1 (Intercept)    0.806    0.446       1.81 0.108        -0.222     1.83
## 2 X              0.794    0.0718     11.1  0.00000400    0.629     0.960
```

From this we see, for example, that the simple linear regression equation is given by:

$$\hat{Y}_i = 0.387 + 0.871 \times X_i,$$

and that we cannot reject $H_0 : \beta_0 = 0$ (p = 0.334), but we strongly reject $H_0 : \beta_1 = 0$ (p = 0.000000536), indicating that a model with no zero intercept will be just as good:

```
fictitious_lm.2 = lm(
   formula = Y ~ -1 + X,
   data = fictitious_data)

# Compare the full model to the reduced model with zero intercept
anova(fictitious_lm, fictitious_lm.2) %>% tidy()
```

```
## # A tibble: 2 x 6
##   res.df   rss    df sumsq statistic p.value
##    <dbl> <dbl> <dbl> <dbl>     <dbl>   <dbl>
## 1      8  3.41    NA NA        NA     NA
## 2      9  4.80    -1 -1.39      3.27   0.108
```

We see that there is no significant reduction in sums of squares between the model with the unconstrained intercept and the model with the intercept constrained to be zero. We therefore opt to use the latter model:

```
tidy(x = fictitious_lm.2,
     conf.int = TRUE,
     conf.level = 0.95)
```

```
## # A tibble: 1 x 7
##   term  estimate std.error statistic       p.value conf.low conf.high
##   <chr>    <dbl>     <dbl>     <dbl>         <dbl>    <dbl>     <dbl>
## 1 X        0.909    0.0372      24.4 0.00000000154    0.825     0.994
```

which has regression equation:

$$\hat{Y}_i = 0.927 \times X_i$$

Using `glance(fictitious_lm.2)` we can obtain a one-row summary of the reduced model. For the linear model this summary contains various statistics about the fit of the model, such as the residual standard error and R^2:

```
# One-row summary at a glance
glance(x = fictitious_lm.2)
```

```
## # 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     0.985         0.984 0.730        NA      NA    NA  -10.5  25.0  25.6
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Using `augment(fictitious_lm.2)` we get back an observation-level data frame containing the original data used to fit the model as well as fitted values (`.fitted`) and standard errors (`.se.fit`).

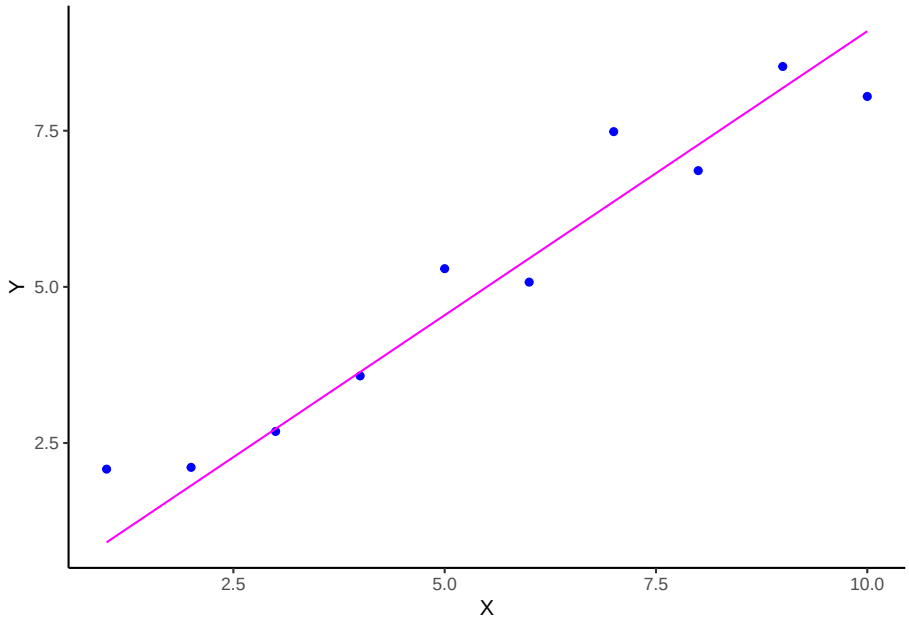
```
augment(x = fictitious_lm.2,
        newdata = NULL,
        se_fit = TRUE,
        interval = "none") # confidence, prediction
```

```
## # A tibble: 10 x 9
##        Y     X .fitted .se.fit   .resid    .hat .sigma   .cooksd .std.resid
##    <dbl> <int>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>     <dbl>      <dbl>
##  1  2.08     1   0.909  0.0372  1.17   0.00260  0.654 0.00673       1.61
##  2  2.11     2   1.82   0.0744  0.291  0.0104   0.768 0.00169       0.401
##  3  2.68     3   2.73   0.112  -0.0434 0.0234   0.774 0.0000867    -0.0602
##  4  3.57     4   3.64   0.149  -0.0631 0.0416   0.774 0.000338     -0.0882
##  5  5.29     5   4.55   0.186   0.745  0.0649   0.725 0.0772        1.05
##  6  5.08     6   5.46   0.223  -0.380  0.0935   0.762 0.0309       -0.547
##  7  7.49     7   6.37   0.261   1.12   0.127    0.648 0.393         1.64
##  8  6.86     8   7.27   0.298  -0.414  0.166    0.758 0.0768       -0.621
##  9  8.53     9   8.18   0.335   0.344  0.210    0.762 0.0751        0.531
## 10  8.05    10   9.09   0.372  -1.04   0.260    0.645 0.969        -1.66
```

Notice that `augment()` also appends model-specific statistics that enable deeper exploration of the fit of the model. For example, we can explore how well the model fits by plotting the observed values `Y` against the fitted values `.fitted` (compare this to the plot constructed earlier, which was based on the full model with unconstrained intercept).

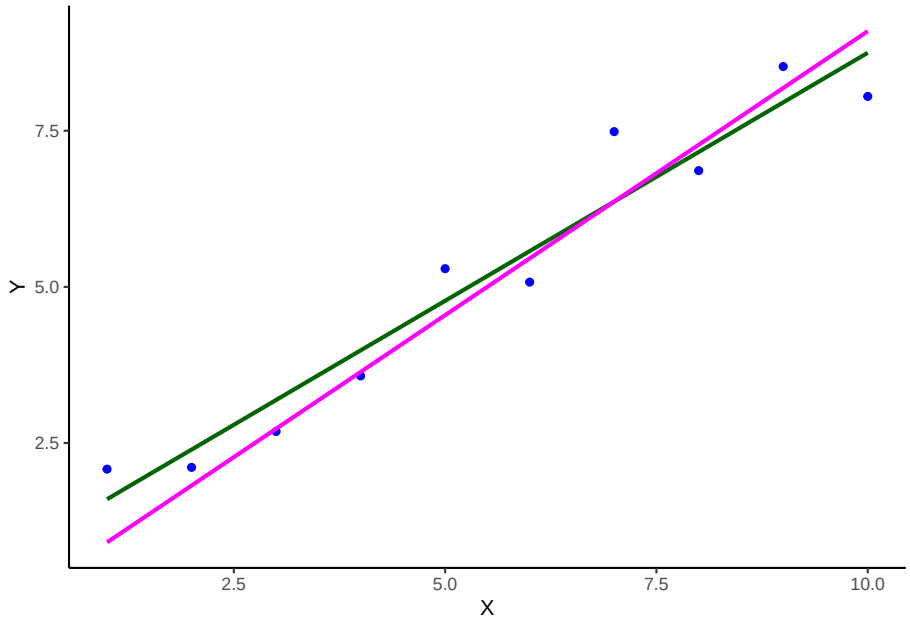
```
p1 = augment(x = fictitious_lm.2) %>%
   ggplot(mapping = aes(x = X)) +
   geom_point(mapping = aes(y = Y), colour="blue") +
   geom_line(mapping = aes(y = .fitted), colour="magenta") +
   theme_classic()

p1
```



We could overlay the unconstrained model as follows:

```
ggplot(data = fictitious_data,
       mapping = aes(x=X, y=Y)) +
   geom_point(colour = "blue") +
   geom_smooth(method = "lm",
               formula = y ~ x,
               se = FALSE,
               colour = "darkgreen") +
   geom_smooth(method = "lm",
               formula = y ~ -1 + x,
               se = FALSE,
               colour = "magenta") +
   theme_classic()
```



4.2 Estimating linear model parameters

During model fitting, parameters can be estimated using a variety of estimation methods. The methods of ordinary least squares (OLS), maximum likelihood (ML) and restricted maximum likelihood (REML) are the most common. The OLS estimates of parameters minimize the sum of squared deviations between the observed and fitted values (a function of the model parameters). There are several variants on the method of ordinary least squares: weighted least squares can accommodate data that varies in quality and generalized least squares can accommodate heterogeneous variances and correlated data. The model we've been looking at is usually referred to as the general linear model (and it was once the most general linear model, but it is not very general by today's standards). Broadly speaking we consider such a model with purely continuous predictor variables to be a **regression** model, while a model with purely categorical predictors is an **analysis of variance (ANOVA)** model. Models that incorporate both continuous and categorical predictors are **analysis of covariance (ANCOVA)** models.

ML estimators estimate model parameters such that the (log) likelihood of obtaining the observed data is maximized. Under standard assumptions (normal data, constant variance, independent observations) the ML estimates are identical to the OLS estimates. However, maximum likelihood parameter estimation extends beyond the general linear model to the so-called "generalized" linear model, which are not restricted to normally distributed errors and response. In-

stead they generalize to any distribution belonging to the family of distribution models called exponential dispersion models (which include normal, binomial, Poisson, gamma, negative binomial and others).

Chapter 5

Applications

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Final Words

We have finished a nice book.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2021). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.22.