Tidy Modeling with R

Max Kuhn and Davis Vaughan

2020-01-09

Contents

Hello World

This is the website for *Tidy Modeling with R*. Its purpose is to be a guide to using a new collection of software in the R programming language that enable model building. There are few goals, depending on your background. First, if you are new to modeling and R, we hope to provide an introduction on how to use our software to create models. The focus will be on a dialect of R called *the tidyverse* that is designed to be a better interface for common tasks using R. If you've never heard of the tidyverse, there is a chapter that provides a solid introduction. The second (and primary) goal is to demonstrate how the tidyverse can be used to produce high quality models. The tools used to do this are referred to as the *tidymodels packages*. The third goal is to use the tidymodels packages to encourage good methodology and statistical practice. Many models, especially complex predictive or machine learning models, can work very well on the data at hand but may also fail when exposed to new data. Often, this issue is due to poor choices that were made during the development and/or selection of the models. Whenever possible, our software attempts to prevent these and other pitfalls.

This book is not intended to be a reference on different types of these techniques We suggest other resources to learn the nuances of models. A general source for information about the most common type of model, the *linear model*, we suggest Fox (2008). Another excellent resource for investigating and analyzing data is Wickham and Grolemund (2016). For predictive models, Kuhn and Johnson (2013) is a good resource. For pure machine learning methods, Goodfellow et al. (2016) is an excellent (but formal) source of information. In some cases, we describe some models that are used in this text but in a way that is less mathematical (and hopefully more intuitive).

We do not assume that readers will have had extensive experience in model building and statistics. Some statistical knowledge is required, such as: random sampling, variance, correlation, basic linear regression, and other topics that are usually found in a basic undergraduate statistics or data analysis course.

Tidy Modeling with R is currently a work in progress. As we create it, this website is updated. Be aware that, until it is finalized, the content and/or structure of the book may change.

This openness also allows users to contribute if they wish. Most often, this comes in the form of correcting typos, grammar, and other aspects of our work that could use improvement. Instructions for making contributions can be found in the `contributing.md` file. Also, be aware that this effort has a code of conduct, which can be found at `code_of_conduct.md`.

In terms of software lifecycle, the tidymodels packages are fairly young. We will do our best to maintain backwards compatibility and, at the completion of this work, will archive the specific versions of software that were used to produce it. The primary packages, and their versions, used to create this website are:

```
#> - Session info ---------------------------------------------------------------
#>  setting  value
#>  version  R version 3.6.1 (2019-07-05)
#>  os       macOS Catalina 10.15.1
#>  system   x86_64, darwin15.6.0
#>  ui       X11
#>  language (EN)
#>  collate  en_US.UTF-8
#>  ctype    en_US.UTF-8
#>  tz       America/New_York
#>  date     2020-01-09
#>
#> - Packages -------------------------------------------------------------------
#>  package     * version    date       lib source
#>  AmesHousing * 0.0.3      2017-12-17 [1] CRAN (R 3.6.0)
#>  bookdown    * 0.16       2019-11-22 [1] CRAN (R 3.6.0)
#>  broom         0.5.3      2019-12-14 [1] CRAN (R 3.6.0)
#>  dplyr       * 0.8.3      2019-07-04 [1] CRAN (R 3.6.0)
#>  ggplot2     * 3.2.1      2019-08-10 [1] CRAN (R 3.6.0)
#>  purrr       * 0.3.3      2019-10-18 [1] CRAN (R 3.6.0)
#>  rlang         0.4.2.9000 2019-12-26 [1] Github (r-lib/rlang@ce4f717)
#>  tibble      * 2.1.3      2019-06-06 [1] CRAN (R 3.6.0)
#>
#> [1] /Library/Frameworks/R.framework/Versions/3.6/Resources/library
```

`pandoc` is also instrumental in creating this work. The version used here is 2.3.1.

Chapter 1

Introduction

Models are mathematical tools that create equations that are intended to mimic the data given to them. These equations can be used for various purposes, such as: predicting future events, determining if there is a difference between several groups, as an aid to a map-based visualization, discovering novel patterns in the data that could be further investigated, and so on. Their utility hinges on their ability to be reductive; the primary influences in the data can be captured mathematically in a way that is useful.

Since the start of the 21st century, mathematical models have become ubiquitous in our daily lives, in both obvious and subtle ways. A typical day for many people might involve checking the weather to see when a good time would be to walk the dog, ordering a product from a website, typing (and autocorrecting) a text message to a friend, and checking email. In each of these instances, there is a good chance that some type of model was used in an assistive way. In some cases, the contribution of the model might be easily perceived ("You might also be interested in purchasing product X") while in other cases the impact was the absence of something (e.g., spam email). Models are used to choose clothing that a customer might like, a molecule that should be evaluated as a drug candidate, and might even be the mechanism that a nefarious company uses avoid the discovery of cars that over-pollute. For better or worse, models are here to stay.

Two reasons that models permeate our lives are that software exists that facilitates their creation and that data has become more easily captured and accessible. In regard to software, it is obviously critical that software produces the *correct* equations that represent the data. For the most part, determining mathematical correctness is possible. However, the creation of an appropriate model hinges on a few other aspects.

First, it is important that it is easy to operate the software in a *proper way*. For example, the user interface should not be so arcane that the user would not know that they have inappropriately specified the wrong information. As an analogy, one might have a high quality kitchen measuring cup capable of great precision but if the chef adds a cup of salt instead of a cup of sugar, the results would be unpalatable. As a specific example of this issue, Baggerly and Coombes (2009) report myriad problems in the data analysis in a high profile computational biology publication. One of the issues was related to how the users were required to add the names of the model inputs. The user-interface of the software was poor enough that it was easy to *offset* the column names of the data from the actual data columns. In the analysis of the data, this resulted in the wrong genes being identified as important for treating cancer patients. This, and many other issues, led to the stoppage of numerous clinical trials (Carlson, 2012).

If we are to expect high quality models, it is important that the software facilitate proper usage. Abrams (2003) describes an interesting principle to live by:

> The Pit of Success: in stark contrast to a summit, a peak, or a journey across a desert to find victory through many trials and surprises, we want our customers to simply fall into winning practices by using our platform and frameworks.

Data analysis software should also espouse this idea.

The second important aspect of model building is related to *scientific methodology.* For models that are used to make complex predictions, it can be easy to unknowingly commit errors related to logical fallacies or inappropriate assumptions. Many machine learning models are so adept at finding patterns, they can effortlessly find empirical patterns in the data that fail to reproduce later. Some of these types of methodological errors are insidious in that the issue might be undetectable until a later time when new data that contain the true result are obtained. In short, as our models become more powerful and complex it has also become easier to commit latent errors. This also relates to programming. Whenever possible, the software should be able to protect users from committing such mistakes. Software should make it easy for users to do the right thing.

These two aspects of model creation are crucial. Since tools for creating models are easily obtained and models can have such a profound impact, many more people are creating them. In terms of technical expertise and training, their backgrounds will vary. It is important that their tools be *robust* to the experience of the user. On one had, they tools should be powerful enough to create high-performance models but, on the other hand, should be easy to use in an appropriate way. This book describes a suite of software that can can create different types of models. The software has been designed with these additional characteristics in mind.

The software is based on the R programming language (R Core Team, 2014). R has been designed especially for data analysis and modeling. It is based on the *S language* which was created in the 1970's to

> "turn ideas into software, quickly and faithfully" (Chambers, 1998)

R is open-source and is provided free of charge. It is a powerful programming language that can be used for many different purposes but specializes in data analysis, modeling, and machine learning. R is easily *extensible*; it has a vast ecosystem of *packages*; these are mostly user-contributed modules that focus on a specific theme, such as modeling, visualization, and so on.

One collection of packages is called the ***tidyverse*** (Wickham et al., 2019). The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures. Several of these design philosophies are directly related to the aspects of software described above. If you've never used the tidy verse packages, Chapter 2 contains a review. Within the tidyverse, there is a set of packages specifically focused on modeling and these are usually referred to as the ***tidymodels*** packages. This book is an extended software manual for conducting modeling using the tidyverse. It shows how to use a set of packages, each with its own specific purpose, together to create high-quality models.

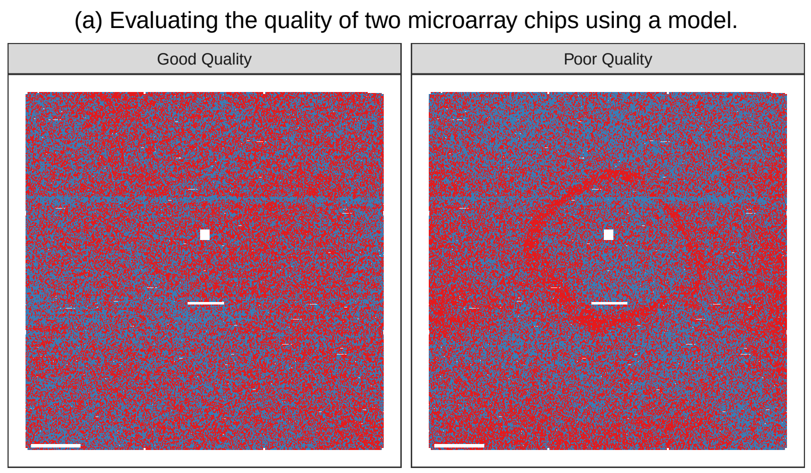
1.1 Types of models

Before proceeding, lets describe a taxa for types of models, grouped by *purpose*. While not exhaustive, most models fail into *at least* one of these categories:

Descriptive Models: The purpose here would be to model the data so that it can be used to describe or illustrate characteristics of some data. The analysis might have no other purpose than to visually emphasize some trend or artifact in the data.

For example, large scale measurements of RNA have been possible for some time using *microarrays*. Early laboratory methods placed a biological sample on a small microchip. Very small locations on the chip would be able to assess a measure of signal based on the abundance of a specific RNA sequence. The chip would contain thousands (or more) outcomes, each a quantification of the RNA related to some biological process. However, there could be quality issued on the chip that might lead to poor results. A fingerprint accidentally left on a portion of the chip might cause inaccurate measurements when scanned.

An early methods for evaluating such issues where *probe-level models*, or PLM's (Bolstad, 2004). A statistical model would be created that accounted for the *known* differences for the data from the chip, such as the RNA sequence, the type of sequence and so on. If there were other, unwanted factors in the data, these would be contained in the model residuals. When the residuals were plotted by their location on the chip, a good quality chip would show no patterns. When an issue did occur, some sort of spatial pattern would be discernible. Often the type of pattern would suggest the underlying issue (e.g. a fingerprint) and a possible solution (wipe the chip off and rescan). Figure 1.1(a) shows an application of this method for two microarrays taken from Gentleman et al. (2005). The images show two different colors; red is where the signal intensity was larger than the model expects while the blue color shows lower than expected values. The left-hand panel demonstrates a fairly random pattern while the right-hand panel shows some type of unwanted artifact.



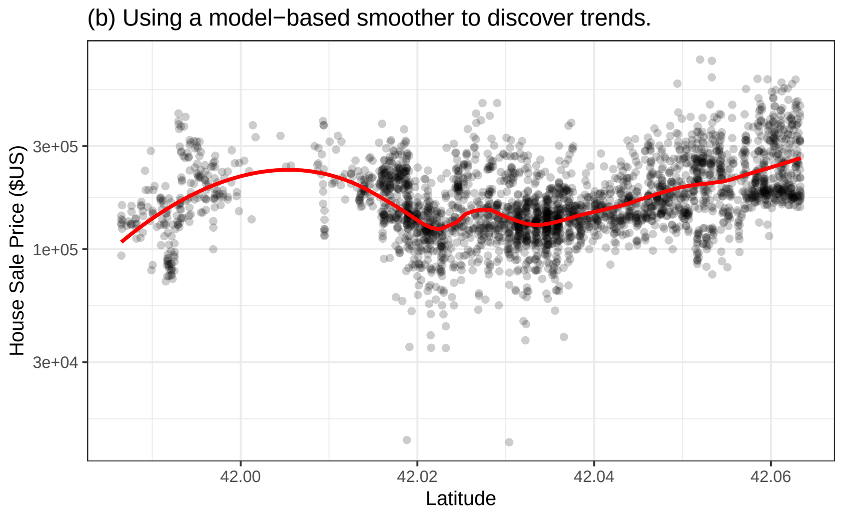


Figure 1.1: Two examples of how descriptive models can be used to illustrate specific patterns.

Another example of a descriptive model is the *locally estimated scatterplot smoothing* model, more commonly known as LOESS (Cleveland, 1979). Here, a smooth and flexible regression model is fit to a data set, usually with a single independent variable, and the fitted regression line is used to elucidate some trend in the data. These types of *smoothers* are used to discover potential ways to represent a variable in a model. This is demonstrated in Figure 1.1(b) where a nonlinear trend is illuminated by the flexible smoother.

Inferential Models: In these situations, the goal is to produce a decision for a research question or to test a specific hypothesis. The goal is to make some statement of truth regarding some predefined conjecture or idea. In many (but not all) cases, some qualitative statement is produced.

For example, in a clinical trial, the goal might be to provide confirmation that a new therapy does a better job in prolonging life than an alternative (e.g., an existing therapy or no treatment). If the clinical endpoint was related to survival or a patient, the *null hypothesis* might be that the two therapeutic groups have equal median survival times with the alternative hypothesis being that the new therapy has higher median survival. If this trial were evaluated using the traditional *null hypothesis significance testing* (NHST), a p-value would be produced using some pre-defined methodology based on a set of assumptions for the data. Small values of the p-value indicate that there is evidence that the new therapy does help patients live longer. If not, the conclusion is that there is a failure to show such an difference (which could be due to a number of reasons).

What are the important aspects of this type of analysis? Inferential techniques typically produce some type of probabilistic output, such as a p-value, confidence interval, or posterior probability. Generally, to compute such a quantity, formal assumptions must be made about the data and the underlying processes that generated the data. The quality of the statistical results are highly dependent on these pre-defined assumptions as well as how much the observed data appear to agree with them. The most critical factors here are theoretical in nature: if my data were independent and follow distribution X, then test statistic Y can be used to produce a p-value. Otherwise, the resulting p-value might be inaccurate.

One aspect of inferential analyses is that there *tends* to be a longer feedback loop that could help understand how well the data fit the assumptions. In our clinical trial example, if statistical (and clinical) significance indicated that the new therapy should be available for patients to use, it may be years before it is used in the field and enough data were generated to have an independent assessment of whether the original statistical analysis led to the appropriate decision.

Predictive Models: There are occasions where data are modeled in an effort to produce the most accurate prediction possible for new data. Here, the primary goal is that the predicted values have the highest possible fidelity to the true value of the new data.

A simple example would be for a book buyer to predict how many copies of a particular book should be shipped to his/her store for the next month. An over-prediction wastes space and money due to excess books. If the prediction is smaller than it should be, there is opportunity loss and less profit.

For this type of model, the problem type is one of *estimation* rather than inference. For example, the buyer is usually not concerned with a question such as “Will I sell more than 100 copies of book X next month?” but rather “How many copies of X will customers purchase next month?” Also, depending on the context, there may not be any interest in *why* the predicted value is X. In other words, is more interest in the value itself than evaluating a formal hypothesis related to the data. That said, the prediction can also include measures of uncertainty. In the case of the book buyer, some sort of forecasting error might be valuable to help them decide on how many to purchase or could serve as a metric to gauge how well the prediction method worked.

What are the most important factors affecting predictive models? There are many different ways that a predictive model could be created. The important factors depend on how the model was developed.

For example, a *mechanistic model* could be developed based on first principles to produce a model equation that is dependent on assumptions. For example, when predicting the amount of a drug that is in a person's body at a certain time, some formal assumptions are made on how the drug is administered, absorbed, metabolized, and eliminated. Based on this, a set of differential equations can be used to derive a specific model equation. Data are used to estimate the known parameters of this equation and predictions are made after parameter estimation. Like inferential models, mechanistic predictive models greatly depend on the assumptions that define their model equations. However, unlike inferential models, it is easy to make data-driven statements about how well the model performs based on how well it predicts the existing data. Here the feedback loop for the modeler is much faster than it would be for a hypothesis test.

Empirically driven models are those that have more vague assumptions that are used to create their model equations. These models tend to fall more into the machine learning category. A good example is the simple K-nearest neighbor (KNN) model. Given a set of reference data, a new sample is predicted by using the values of the most similar data in the reference set. For example, if a book buyer needs a prediction for a new book, historical data from existing books may be available. A 5-nearest neighbor model would estimate the amount of the new book to purchase based on the sales numbers of the five books that are most similar to the new one (for some definition of "similar"). This model is only defined by the structure of the prediction (the average of five similar books). No theoretical or probabilistic assumptions are made about the sales numbers or the variables that are used to define similarity. In fact, the primary method of evaluating the appropriateness of the model is to assess its accuracy using existing data. If the structure of this type of model was a good choice, the predictions would not be close to the actual values.

Broader discussions of these distinctions can be found in Breiman (2001) and Shmueli (2010). Note that we have defined the type of model by how it is used rather than its mathematical qualities. An ordinary linear regression model might fall into all three classes of models, depending on how it is used:

- Descriptive smoother, similar to LOESS, called *restricted smoothing splines* (Durrleman and Simon, 1989) can be used to describe trends in data using ordinary linear regression with specialized terms.
- An *analysis of variance* (ANOVA) model is a popular method for producing the p-values used for inference. ANOVA models are a special case of linear

regression.

- If a simple linear regression model produces highly accurate predictions, it can be used as a predictive model.

However, there are many more examples of predictive models that cannot (or at least should not) be used for inference. Even if probabilistic assumptions were made for the data, the nature of the KNN model makes the math required for inference intractable.

There is an additional connection between the types of models. While the primary purpose of descriptive and inferential models might not be related to prediction, the predictive capacity of the model should not be ignored. For example, logistic regression is a popular model for data where the outcome is qualitative with two possible values. It can model how variables related to the probability of the outcomes. When used in an inferential manner, there is usually an abundance of attention paid to the *statistical qualities* of the model. For example, analysts tend to strongly focus on the selection of which independent variables are contained in the model. Many iterations of model building are usually used to determine a minimal subset of independent variables that have a "statistically significant" relationship to the outcome variable. This is usually achieved when all of the p-values for the independent variables are below some value (e.g. 0.05). From here, the analyst typically focuses on making qualitative statements about the relative influence that the variables have on the outcome.

A potential problem with this approach is that it can be dangerous when statistical significance is used as the *only* measure of model quality. It is certainly possible that this statistically optimized model has poor model accuracy (or some other measure of predictive capacity). While the model might not be used for prediction, how much should the inferences be trusted from a model that has all significant p-values but a dismal accuracy? Predictive performance tends to be related to how close the model's fitted values are to the observed data. If the model has limited fidelity to the data, the inferences generated by the model should be highly suspect. In other words, statistical significance may not imply that the model should be used. This may seem intuitively obvious, but is often ignored in real-world data analysis.

1.2 Some terminology

Before proceeding, some additional terminology related to modeling, data, and other quantities should be outlined. These descriptions are not exhaustive.

First, many models can be categorized as being *supervised* or *unsupervised*. Unsupervised models are those that seek patterns, clusters, or other characteristics of the data but lack an outcome variable (i.e., a dependent variable). For example, principal component analysis (PCA), clustering, and autoencoders are used to understand relationships between variables or sets of variables without an explicit relationship between variables and an outcome. Supervised models are those that have an outcome variable. Linear regression, neural networks, and numerous other methodologies fall into this category. Within supervised models, the two main sub-categories are:

- *Regression*, where a numerical outcome is being predicted.
- *Classification*, where the outcome is an ordered or unordered set of *qualitative* values.

These are imperfect definitions and do not account for all possible types of models. In coming chapters, we refer to these types of supervised techniques as the *model mode*.

In terms of data, the main species are quantitative and qualitative. Examples of the former are real numbers and integers. Qualitative values, also known as nominal data, are those that represent some sort of discrete state that cannot be placed on a numeric scale.

Different variables can have different *roles* in an analysis. Outcomes (otherwise known as the labels, endpoints, or dependent variables) are the value being predicted in supervised models. The independent variables, which are the substrate for making predictions of the outcome, also referred to as predictors, features, or covariates (depending on the context). Here, the terms *outcomes* and *predictors* are used most frequently here.

1.3 How does modeling fit into the data analysis/scientific process?

In what circumstances are model created? Are there steps that precede such an undertaking? Is it the first step in data analysis?

There are always a few critical phases of data analysis that come before modeling. First, there is the chronically underestimated process of **cleaning the data**. No matter the circumstances, the data should be investigated to make sure that it is well understood, applicable to the project goals, accurate, and appropriate. These steps can easily take more time than the rest of the data analysis process (depending on the circumstances).

Data cleaning can also overlap with the second phase of **understanding the data**, often referred to as exploratory data analysis (EDA). There should be knowledge of how the different variables related to one another, their distributions, typical ranges, and other attributes. A good question to ask at this phase is "how did I come by *these* data?" This question can help understand how the data at-hand have been sampled or filtered and if these operations were appropriate. For example, when merging data base tables, a join may go awry that could accidentally eliminate one or more sub-populations of samples. Another good idea would be to ask if the data are *relavant*. For example, to predict whether patients have Alzheimer's disease or not, it would be unwise to have a data set containing subject with the disease and a random sample of healthy adults from the general population. Given the progressive nature of the disease, the model my simply predict who the are the *oldest patients*.

Finally, before starting a data analysis process, there should be clear expectations of the goal of the model and how performance (and success) will be judged. At least one *performance metric* should be identified with realistic goals of what can be achieved. Common statistical metrics are classification accuracy, true and false positive rates, root mean squared error, and so on. The relative benefits and drawbacks of these metrics should be weighted. It is also important that the metric be germane (i.e., alignment with the broader data analysis goals is critical).

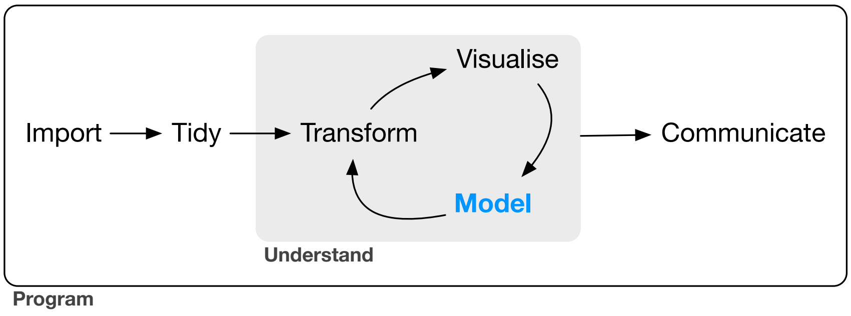


Figure 1.2: The data science process (from R for Data Science).

The process of investigating the data may not be simple. Wickham and Grolemund (2016) contains an excellent illustration of the general data analysis process, reproduced with Figure 1.2. Data ingestion and cleaning are shown as the initial steps. When the analytical steps commence, they are a heuristic process; we cannot pre-determine how long they may take. The cycle of analysis, modeling, and visualization often require multiple iterations.

This iterative process is especially true for modeling. Figure 1.3 originates from Kuhn and Johnson (2020) and is meant to emulate the typical path to determining an appropriate model. The general phases are:

- Exploratory data analysis (EDA) and Quantitative Analysis (blue bars). Initially there is a back and forth between numerical analysis and visualization of the data (represented in Figure 1.2) where different discoveries lead to more questions and data analysis "side-quests" to gain more understanding.

- Feature engineering (green bars). This understanding translated to the creation of specific model terms that make it easier to accurately model the observed data. This can include complex methodologies (e.g., PCA) or simpler features (using the ratio of two predictors).

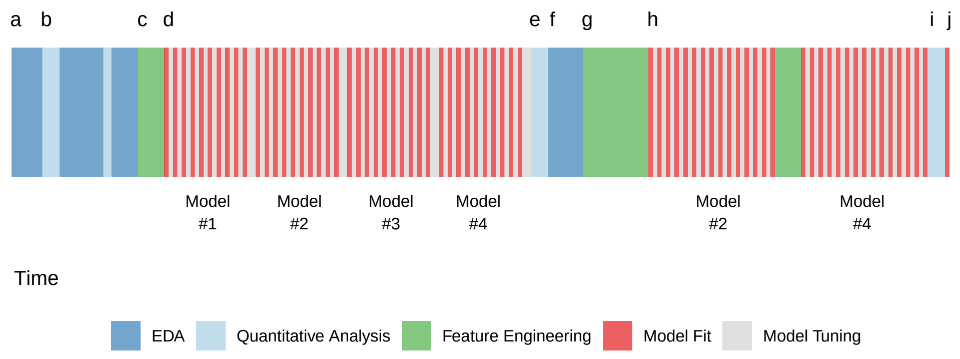


Figure 1.3: A schematic for the typical modeling process (from Feature Engineering and Selection).

- Model tuning and selection (red and gray bars). A variety of models are generated and their performance is compared. Some models require *parameter tuning* where some structural parameters are required to be specified or optimized.

After an initial sequence of these tasks, more understanding is gained regarding which types of models are superior as well as which sub-populations of the data are not being effectively estimated. This leads to additional EDA and feature engineering, another round of modeling, and so on. Once the data analysis goals are achieved, the last steps are typically to finalize and document the model. For predictive models, it is common at the end to validate the model on an additional set of data reserved for this specific purpose.

1.4 Where does the model begin and end?

So far, we have defined the model to be a structural equation that relates some predictors to one or more outcomes. Let's consider ordinary linear regression as a simple and well known example. The outcome data are denoted as y_i, where there are $i = 1 \ldots n$ samples in the data set. Suppose that there are p predictors $x_{i1}, \ldots, x_{ip}$ that are used to predict the outcome. Linear regression produces a model equation of

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \ldots + \hat{\beta}_p x_{ip}$$

While this is a *linear* model, it is only linear in the parameters. The predictors could be nonlinear terms (such as the $log(x_i)$).

The conventional way of thinking is that the modeling *process* is encapsulated by the model. For many data sets that are straight-forward in nature, this is the case.

However, there are a variety of *choices* and additional steps that often occur before the data are ready to be added to the model. Some examples:

- While our model has p predictors, it is common to start with more than this number of candidate predictors. Through exploratory data analysis or previous experience, some of the predictors may be excluded from the analysis. In other cases, some feature selection algorithm may have been used to make a data-driven choice for the minimum predictors set to be used in the model.
- There are times when the value of an important predictor is know known. Rather than eliminating this value from the data set, it could be *imputed* using other values in the data. For example, if x_1 were missing but was correlated with predictors x_2 and x_3, an imputation method could estimate the missing x_1 observation from the values of x_2 and x_3.
- As previously mentioned, it may be beneficial to transform the scale of a predictor. If there is **not** *a priori* information on what the new scale should be, it might be estimated using a transformation technique. Here, the existing data would be used to statistically *estimate* the proper scale that optimizes some criterion. Other transformations, such as the previously mentioned PCA, take groups of predictors and transform them into new features that are used as the predictors.

While the examples above are related to steps that occur before the model, there may also be operations that occur after the model is created. For example, when a classification model is created where the outcome is binary (e.g., `event` and `non-event`), it is customary to use a 50% probability cutoff to create a discrete class prediction (also known as a "hard prediction"). For example, a classification model might estimate that the probability of an event was 62%. Using the typical default, the hard prediction would be `event`. However, the model may need to be more focused on reducing false positive results (i.e., where true non-events are classified as events). One way to do this is to *raise* the cutoff from 50% to some greater value. This increases the level of evidence required to call a new sample as an event. While this reduces the true positive rate (which is bad), it may have a more profound effect on reducing false positives. The choice of the cutoff value should be optimized using data. This is an example of a post-processing step that has a significant effect on how well the model works even though it is not contained in the model fitting step.

These examples have a common characteristic of requiring data for derivations that alter the raw data values or the predictions generated by the model.

It is very important to focus on the broader *model fitting process* instead of the specific model being used to estimate parameters. This would include any pre-processing steps, the model fit itself, as well as potential post-processing activities. In this text, this will be referred to as the **model workflow** and would include any data-driven activities that are used to produce a final model equation.

This will come into play when topics such as resampling (Chapter 8) and model tuning are discussed. Chapter 7 describes software for creating a model workflow.

1.5 Outline of future chapters

The first order of business is to introduce (or review) the ideas and syntax of the tidyverse in Chapter 2. In this chapter, we also summarize the unmet needs for modeling when using R. This provides good motivation for why model-specific tidyverse techniques are needed. This chapter also outlines some additional principles related to this challenges.

Chapter 3 shows two different data analyses for the same data where one is focused on prediction and the other is for inference. This should illustrates the challenges for each approach and what issues are most relavant for each.

Chapter 2

A tidyverse primer

2.1 Principles

What does it mean to be "tidy" (distinguish tidy data vs tidy interfaces etc.)

2.2 Code

Things that I think that we'll need summaries of:

- strategies: variable specification, pipes (with data or other first arguments), conflicts and using namespaces, splicing, non-standard evaluation,
- tactics: `select`, `bind_cols`, `tidyselect`, `slice`, `!!` and `!!!`, `...` for passing arguments, tibbles, joins, `nest`/`unnest`, `group_by`

2.3 A review of base R modeling syntax

This book is about software, specifically R syntax for creating models. Before descrbing how tidy principles can be used in data analysis, it makes sense to show how models are created and utilized using traditional base R code. This section is a brief illustration of the those conventions. It is not exhaustive but provides readers uninitiated to R ideas about the basic motifs that are commonly used.

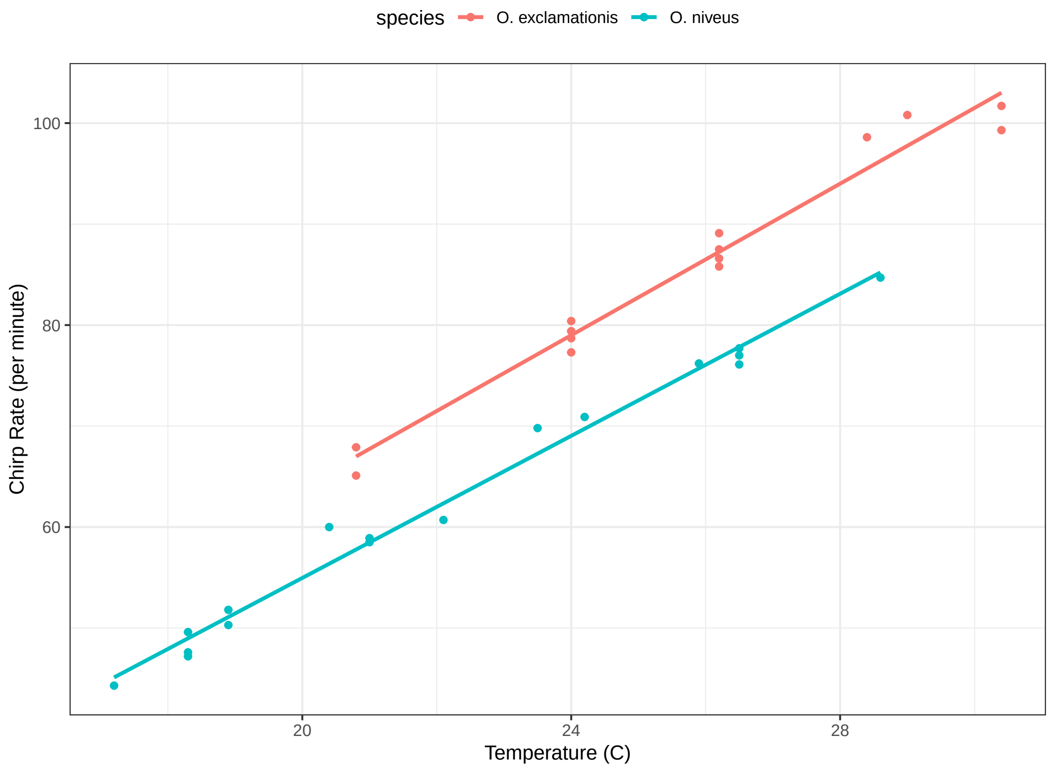
The S language, on which R is based, has had a rich data analysis environment since the publication of Chambers and Hastie (1992) (commonly known as The White Book). This version of S introduced standard infrastructure components, such as symbolic model formulae, model matrices, data frames, as well as the standard object-oriented programming methods for data analysis. Much of these implementations have not substantively changes since then.

To demonstrate the fundamentals, experimental data from McDonald (2009) (by way of Mangiafico (2015)) are used. These data relate how the ambient temperature related to the rate of cricket chirps per minute. Data were collected for two species: *O. exclamationis* and *O. niveus*. The data are contained in a data frame called `crickets` that contains a total of 31 data points. These data are shown via a `ggplot` graph.

```r
library(ggplot2)

names(crickets)
#> [1] "species" "temp"    "rate"

# Plot the temperature on the x-axis, the chirp rate on the y-axis. The plot
# elements will be colored differently for each species:
ggplot(crickets, aes(x = temp, y = rate, col = species)) +
  # Plot points for each data point and color by species
  geom_point() +
  # Show a simple linear model fit created separately for each species:
  geom_smooth(method = lm, se = FALSE) +
  labs(x = "Temperature (C)", y = "Chirp Rate (per minute)")
```



The data show fairly linear trends for each species. For a given temperature, *O. exclamationis* appears to have more chirps than the other species. For an inferential model, the researchers might have specified the following null hypotheses prior to seeing the data:

- Temperature has no affect on the chirp rate (denoted as hypothesis #1)
- There are no differences between the species in terms of chirp rate.

There may be some scientific rationale for being able to predict the chirp rate but the focus here will be on inference.

To fit an ordinary linear model, the `lm()` function is commonly used. The important arguments to this function are a model formula and a data frame that contains the data The formula is *symbolic*. For example, the simple formula:

```
rate ~ temp
```

states that the chirp rate is the outcome (since it is on the left-hand side of the tidle `~`) and that the temperature values are the predictor[1]. Suppose the data contained the time of day in which the measurements were obtained in a column called `time`. The formula

```
rate ~ temp + time
```

would not add the time and temperature values together. This formula would symbolically represent that temperature and time should be added as a separate *main effects* to the model. Main effects are model terms that contain a single predictor variable.

There are no time measurements in these data but the species can be added to the model in the same way:

```
rate ~ temp + species
```

Species is not a quantitative variable; in the data frame, it is represented as a factor column with levels `"O. exclamationis"` and `"O. niveus"`. The vast majority of model functions cannot operate on non-numeric data. For species, the model needs to *encode* the species data into a numeric format. The most common approach is to use indicator variables (also known as “dummy variables”) in place of the original qualitative values. In this instance, since species has two possible values, the model formula will automatically encode this column as numeric by adding a new column that has a value of zero when the species is `"O. exclamationis"` and a value of one when the data correspond to `"O. niveus"`. The underlying formula machinery will automatically convert these data for the data used to create the model as well as for any new data points (for example, when the model is used for prediction).

[1]Most model functions implicitly add an intercept column.

Suppose there were five species. The model formula would automatically add *four* additional binary columns that are binary indicators for four of the species. The *reference level* of the factor (i.e., the first level) is always left out of the predictor set. The idea is that, if you know the values of the four indicator variables, the value of the species can be determined.

The model formula shown above creates a model where there are different y-intercepts for each species. It is a reasonable supposition that the slopes of the regression lines could be different for each species. To accommodate this structure, an *interaction* term can be added to the model. This can be specified in a few different ways, the most basic uses the colon:

```
rate ~ temp + species + temp:species

# A shortcut can be used to expand all interactions containing
# interactions with two variables:
rate ~ (temp + species)^2
```

In addition to the convenience of automatically creating indicator variables, the formula offers a few other niceties:

- *In-line* functions can be used in the formula. For example, if the natural log of the temperate were used, the formula `rate ~ log(temp)` could be used. Since the formula is symbolic by default, literal math can be done to the predictors using the identity function `I()`. For example, to use Fahrenheit units, the formula could be `rate ~ I( (temp * 9/5) + 32 )` to make the conversion.
- R has many functions that are useful inside of formulas. For example, `poly(x, 3)` would create linear, quadratic, and cubic terms for `x` to the model as main effects. Also, the `splines` package has several functions to create nonlinear spline terms in the formula.
- For data sets where there are many predictors, the period shortcut is available. The period represents main effects for all of the columns that are not on the left-hand side of the tilde. For example, using `~ (.)^3` would create main effects as well as all two- and three-variable interactions to the model.

For the initial data analysis, the two-factor interaction model is used. In this book, the suffix `_fit` is used for R objects for fitted models.

```
interaction_fit <-  lm(rate ~ (temp + species)^2, data = crickets)

# To print a short summary of the model:
interaction_fit
#> 
#> Call:
#> lm(formula = rate ~ (temp + species)^2, data = crickets)
```

```
#>
#> Coefficients:
#>          (Intercept)                    temp
#>              -11.041                   3.751
#>      speciesO. niveus  temp:speciesO. niveus
#>               -4.348                  -0.234
```

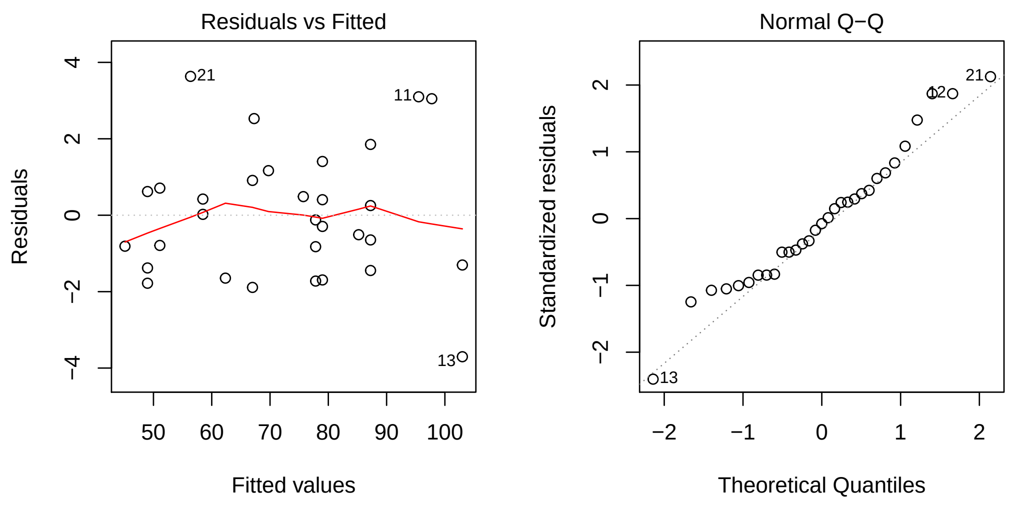
This output is a little hard to read. For the species indicator variables, R mashes the variable name (`species`) together with the factor level (`O. niveus`) with no delimiter.

Before going into any results for this model, the fit should be assessed using diagnostic plots. The `plot()` method for `lm` objects can be used. It produces a set of four plots for the object, each showing different aspects of the fit. Two plots are shown here:

```
# Place two plots next to one another:
par(mfrow = c(1, 2))

# Show residuals vs predicted values:
plot(interaction_fit, which = 1)

# A normal quantile plot on the residuals:
plot(interaction_fit, which = 2)
```



These appear reasonable enough to conduct inferential analysis.

From a technical standpoint, R is *lazy*. Model fitting functions typically compute the minimum possible quantities. For example, there may be interest in the coefficient table for each model term. This is not automatically computed but is instead computed via the `summary()` method.

Our second order of business is to assess if the inclusion of the interaction term is necessary. The most appropriate approach for this model is to re-compute the model without the interaction term and use the `anova()` method.

```
# Fit a reduced model:
main_effect_fit <-  lm(rate ~ temp + species, data = crickets)

# Compare the two:
anova(main_effect_fit, interaction_fit)
#> Analysis of Variance Table
#>
#> Model 1: rate ~ temp + species
#> Model 2: rate ~ (temp + species)^2
#>   Res.Df  RSS Df Sum of Sq    F Pr(>F)
#> 1     28 89.3
#> 2     27 85.1  1      4.28 1.36   0.25
```

The results of the statistical test generates a p-value of 0.3. This value implies that there is a lack of evidence for the alternative hypothesis that the the interaction term is needed by the model. For this reason, further analysis will be conducted on the model without the interaction.

Residual plots should be re-assessed to make sure that our theoretical assumptions are valid enough to trust the p-values produced by the model (not shown but spoiler alert: they are).

The `summary()` method is used to inspect the coefficients, standard errors, and p-values of each model term:

```
summary(main_effect_fit)
#>
#> Call:
#> lm(formula = rate ~ temp + species, data = crickets)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -3.013 -1.130 -0.391  0.965  3.780
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)       -7.2109     2.5509   -2.83   0.0086
#> temp               3.6028     0.0973   37.03  < 2e-16
#> speciesO. niveus -10.0653     0.7353  -13.69  6.3e-14
#>
#> (Intercept)      **
#> temp             ***
#> speciesO. niveus ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.79 on 28 degrees of freedom
#> Multiple R-squared:  0.99,	Adjusted R-squared:  0.989
#> F-statistic: 1.33e+03 on 2 and 28 DF,  p-value: <2e-16
```

From these values, the chirp rate for each species increases by 3.6 chirps as the temperature increases by a single degree. This term shows strong statistical significance as evidenced by the p-value. The species term has a value of -10.07. This indicates that, across all temperature values, *O. niveus* is a chirp rate that is about 10 fewer chirps per minute. Similar to the temperature term, the species effect is associated with a very small p-value.

The only issue in this analysis is the intercept value. It indicates that at 0 C, there are -7.21 chirps per minute. While this is unreasonable, the data only go as low as 17.2 C and interpreting the model at 0 C would be an *extrapolation*. This would be a bad idea. That said, the model fit is good within the *applicable range* of the temperature values and the conclusions should be limited to the observed temperature range.

If there were a need to estimate the chirp rate at a temperature that was not observed in the experiment, the `predict()` method would be used. It takes the model object and a data frame of new values for prediction. For example, the model estimates the chirp rate for *O. exclamationis* for temperatures between 15 C and 20 C can be computed via:

```
new_values <- data.frame(species = "O. exclamationis", temp = 15:20)
predict(main_effect_fit, new_values)
#>    1    2    3    4    5    6
#> 46.8 50.4 54.0 57.6 61.2 64.8
```

Note that the non-numeric value of `species` is given to the predict method (as opposed to the binary indicator variable).

While this analysis has obviously not been an exhaustive demonstration of R's modeling capabilities, it does highlight some of the major features:

- The language has an expressive syntax for specifying model terms for simple and fairly complex models.
- For formula method has many conveniences for modeling that are also applied to new data when predictions are generated.
- There are numerous helper functions (e.g., `anova()`, `summary()` and `predict()`) that are used to conduct specific calculations after the fitted model is created.

Finally, as previously mentioned, this framework was devised in 1992. Most of the ideas and methods above were developed in that period and have remained remarkably relavant to this day. It highlights that the S language and, by extension R, has been designed as a language for data analysis since its inception.

2.4 Why tidiness is important for modeling

2.5 Some additional tidy principals for modeling.

Chapter 3

A tale of two models

(tentative title)

Perhaps show an example of a predictive model and contrast it with another that is inferential.

Chicago data from FES: one predictive model and one to test if there is a difference in ridership with the Bears are at home. what do we care about for each? how accurate is the inferential model? Perhaps look at the `tscount` package to deal with the autoregressive potential.

Chapter 4

Spending our data

General data splitting

Re-emphasize roles or different data sets and good/bad ways of doing things.

Validation sets.

What we do differently with a lot of data.

Allude to resampling.

Chapter 5

How good is our model?

(or how well does our model work? Superman does good; a model can work well)

Measuring performance

Don’t revaluate the training set

Statistical significance as a measure of effectiveness.

Chapter 6

Feature engineering

Purpose(s) of these activites.

Why do we do this?

Different representations of same data

Imputation; transformations; (unsup) removal; projection; encodings;

Chapter 7

A model workflow

aka modeling process or model pipeline

How to encapsulate the pre-processing and model objects/activities

Treat them as a single unit for good methodology and convenience.

Chapter 8

Resampling for evaluating performance

Maybe inlcude some simple examples of comparing models using resampling (perhaps go full `tidyposterior`?)

Bibliography

Abrams, B. (2003). The pit of success. https://blogs.msdn.microsoft.com/brada/2003/10/02/the-pit-of-success/. Accessed: 2019-12-12.

Baggerly, K. and Coombes, K. (2009). Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *The Annals of Applied Statistics*, 3(4):1309–1334.

Bolstad, B. (2004). *Low-level analysis of high-density oligonucleotide array data: Background, normalization and summarization*. University of California, Berkeley.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231.

Carlson, B. (2012). Putting oncology patients at risk. *Biotechnology Healthcare*, 9(3):17–21.

Chambers, J. (1998). *Programming with Data: A Guide to the S Language*. Springer-Verlag, Berlin, Heidelberg.

Chambers, J. and Hastie, T., editors (1992). *Statistical Models in S*. CRC Press, Inc., Boca Raton, FL.

Cleveland, W. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

Durrleman, S. and Simon, R. (1989). Flexible regression models with cubic splines. *Statistics in Medicine*, 8(5):551–561.

Fox, J. (2008). *Applied Regression Analysis and Generalized Linear Models*. Sage, Thousand Oaks, CA, second edition.

Gentleman, R., Carey, V., Huber, W., Irizarry, R., and Dudoit, S. (2005). *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer-Verlag, Berlin, Heidelberg.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Kuhn, M. and Johnson, K. (2020). *Feature engineering and selection: A practical approach for predictive models.* CRC Press.

Mangiafico, S. (2015). An R companion for the handbook of biological statistics. https://rcompanion.org/handbook/. Accessed: 2019-12-22.

McDonald, J. (2009). *Handbook of Biological Statistics.* Sparky House Publishing.

R Core Team (2014). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Shmueli, G. (2010). To explain or to predict? *Statistical science*, 25(3):289–310.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K., Ooms, J., Robinson, D., Seidel, D., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, 4(43).

Wickham, H. and Grolemund, G. (2016). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* O'Reilly Media, Inc.