STAT 1291: Data Science

Lecture 16 - Statistical modeling I: Regression

Sungkyu Jung

Where are we?

- *data visualization*
- *data wrangling*
- *professional ethics*
- *statistical foundation*
- Statistical modeling

Conditional modeling

- A travel policy using the `SF` data (below) as the population.
- Example: book a flight that is scheduled to arrive at least 124 minutes before.
- There, we assumed that the population is *homogeneous*; that is, a flight is just like any other flight.

Heterogeneous population

The assumption of homogeneous population is certainly not true:

- flight delays are more likely to occur if the airport is busy,
- depends on the time of day, day of the week and month of the year.
- Accounting the heterogeneous nature of population is done by modeling.
- A model, explaining a variable (arrival delay) by other variables (time of the year), is necessarily a conditional statement:

- "arrival delay pattern depends on time of the year".

Models

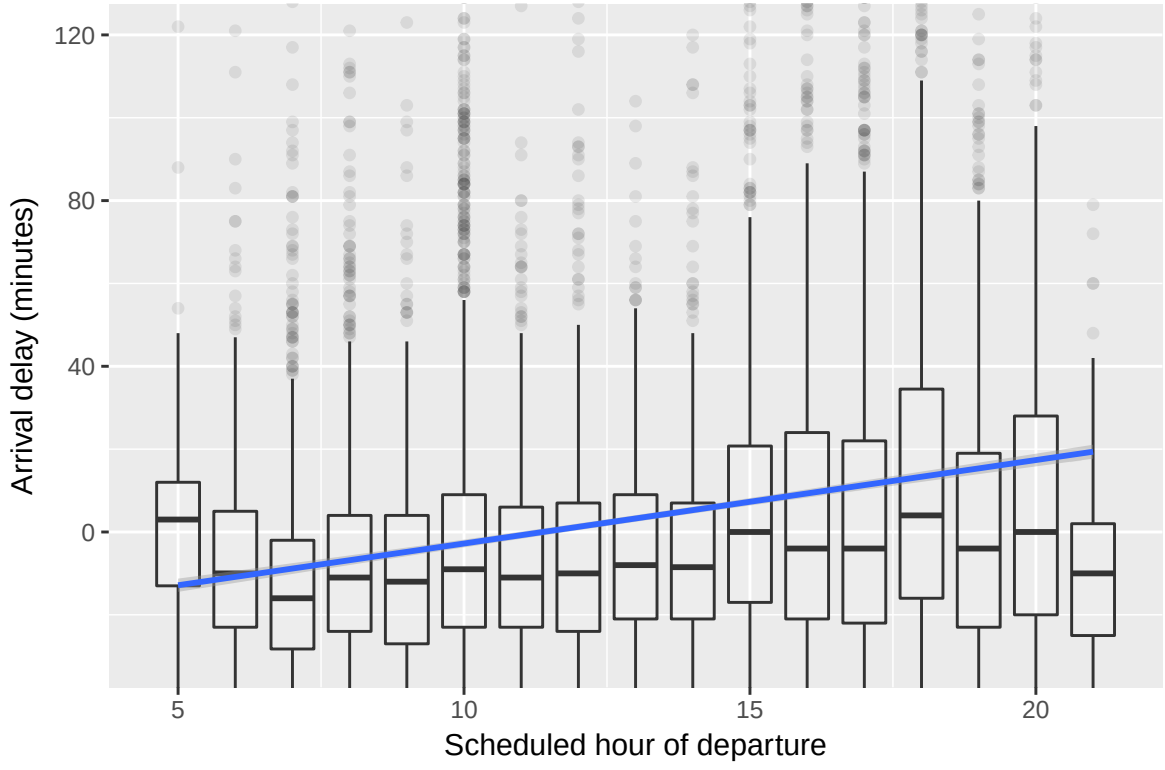
Many of the "useful" models, in statistics, statistical learning, machine learning, or data mining, are categorized into one of the following:

1. **Linear models** are a simple, yet powerful, class of models. Linear models include the models behind t-test, ANOVA, and multiple linear regression.
2. **Generalized linear models** are a larger class of models, enjoying both the interpretability of simple linear models and the power to model more complex relation. Logistic regression is an example.
3. In statistics, **nonlinear models** refer to a class of specific parametric models. There are more drawbacks than advantages in using these models. For example, these are not very flexible, yet computationally expensive to fit.
4. **Nonparametric models** are a vast class of flexible models, including both linear and nonlinear models. Examples include scatterplot smoothers and decision trees.

Toward modeling the arrival delay time

We begin by considering time of day: first using standard box-and-whiskers to show the (conditional) distribution of arrival delays per each hour; second with a kind of statistical model called a linear model that lets us track the mean arrival delay over the course of the day.

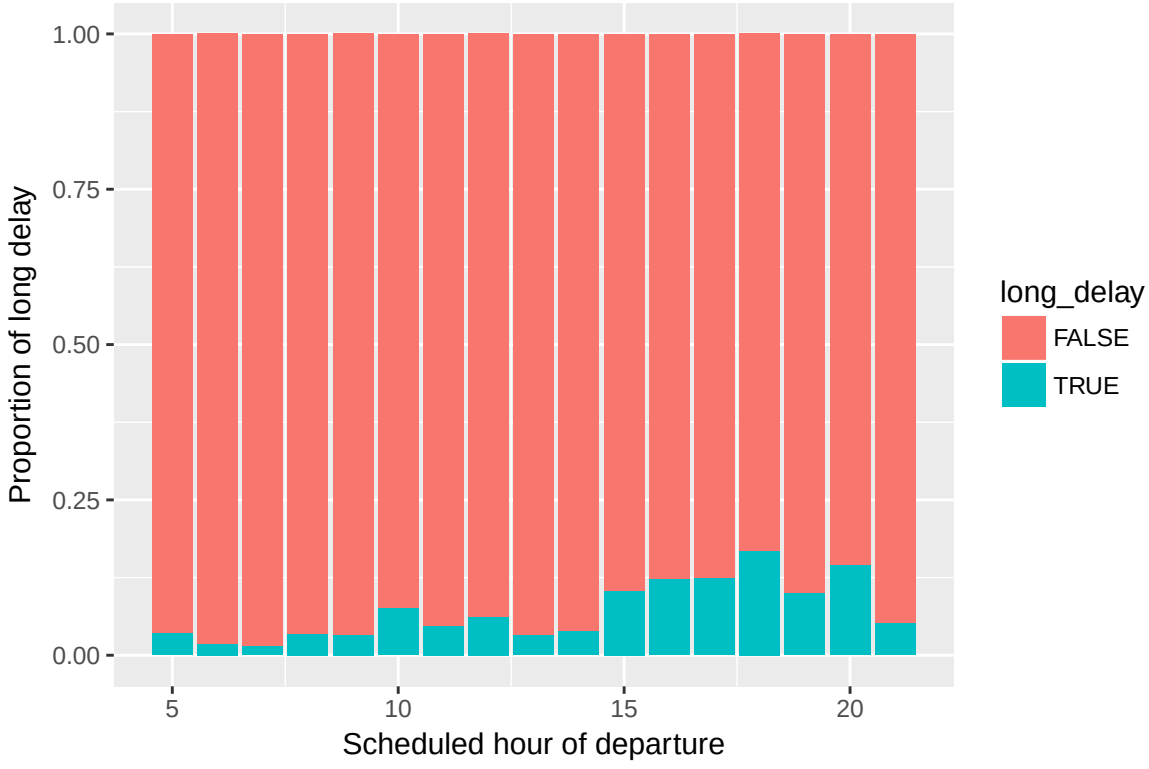
```
SF %>%
  ggplot(aes(x = hour, y = arr_delay)) +
  geom_boxplot(alpha = 0.1, aes(group = hour)) +
  geom_smooth(method = "lm") +
  xlab("Scheduled hour of departure") + ylab("Arrival delay
```



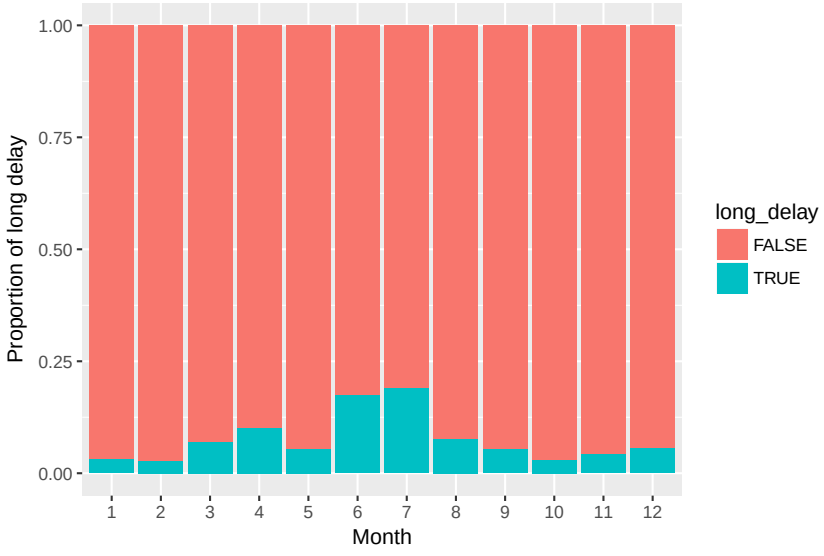
Data transformation might help us better understand the pattern. For this, create a new variable `long_delay` whose value is 1 `TRUE` if the delay is over 60 minutes. Instead of looking at the whole distribution of arrival delay, focus on the binary variable `long_delay`.

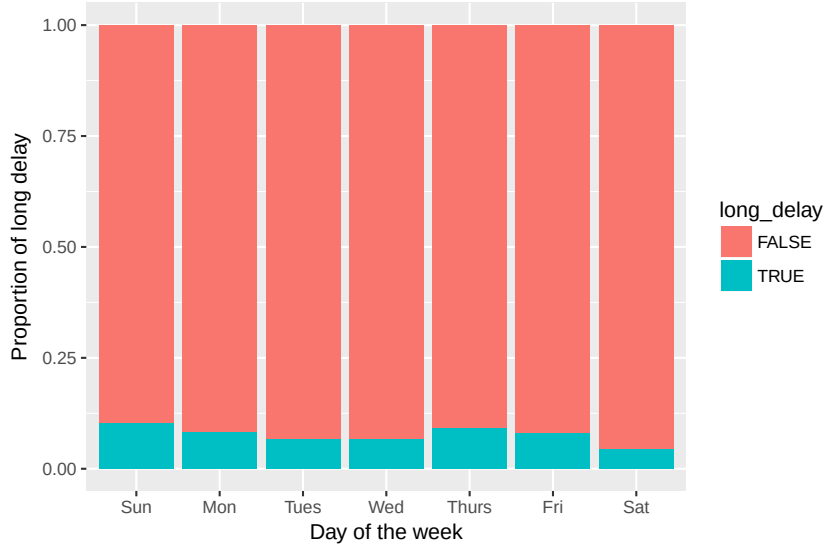
```
SF %>%
  mutate(long_delay = arr_delay > 60)
```

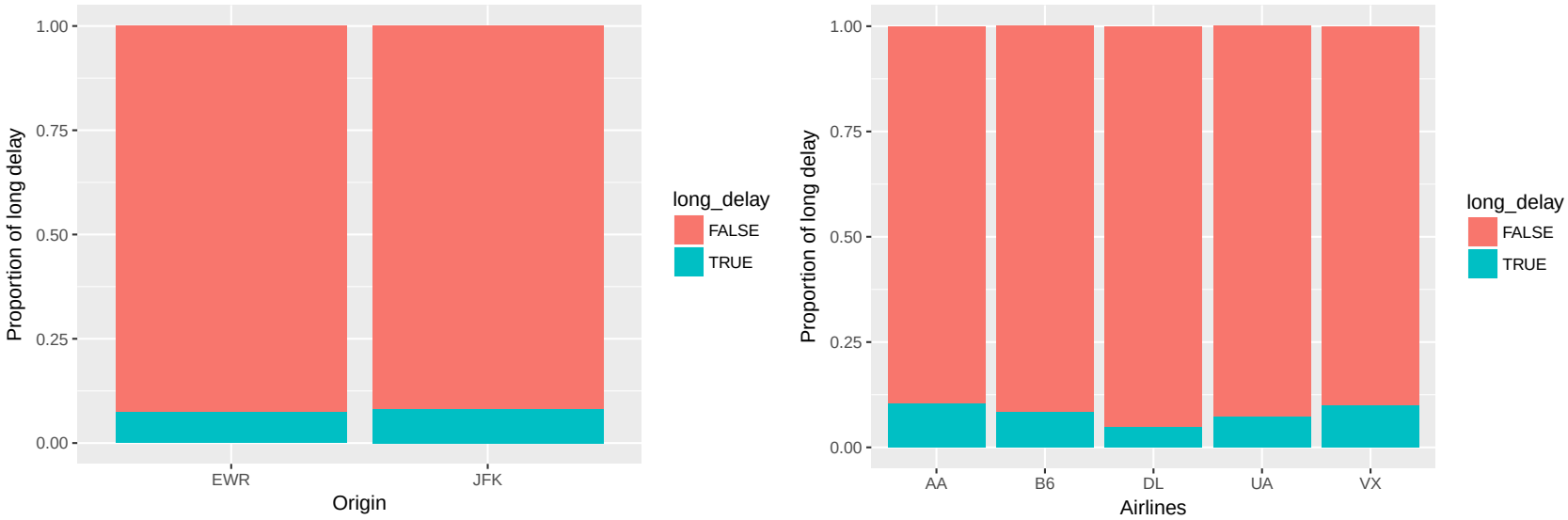
```
SF %>%
  mutate(long_delay = arr_delay > 60) %>%
  ggplot(aes(x = hour, fill= long_delay)) +
  geom_bar(position = "fill") +
  xlab("Scheduled hour of departure") + ylab("Proportion of
```



How about in terms of day of the week, month of the year, and different airlines?






1.00
0.75
0.50
0.25
0.00
Proportion of long delay
EWR
JFK
Origin
long_delay
FALSE
TRUE
AA
B6
DL
UA
VX
Airlines

It makes a lot of sense to use these variables to model the patterns of arrival delay.

- Is the visual pattern really there?
- Can modeling help us identifying patterns that are not easily visible?

Linear regression

We use a different data set `mosaicData::RailTrail` to make examples for regression. The Pioneer Valley Planning Commission (PVPC) collected data north of Chestnut Street in Florence, Massachusetts for a ninety day period.

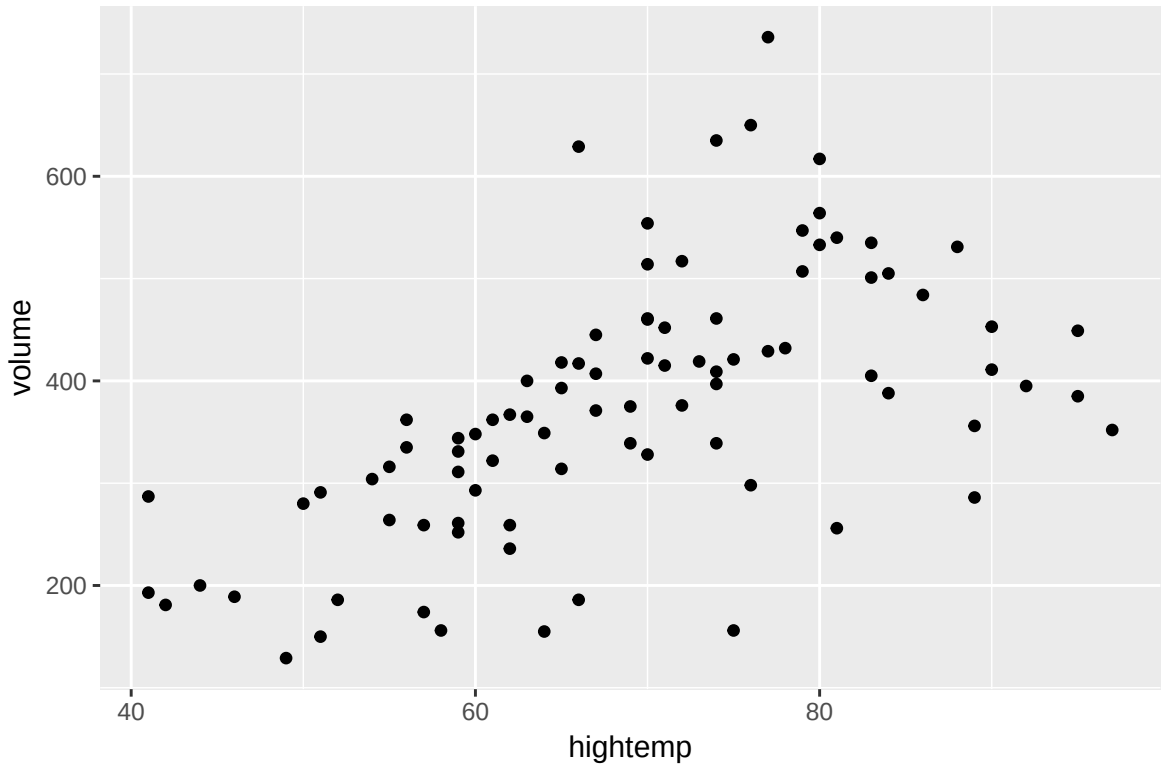
The PVPC wants to understand the relationship between daily ridership (i.e., the number of riders and walkers who use the bike path on any given day) and a collection of explanatory variables, including the temperature, rainfall, cloud cover, and day of the week.

```
## Observations: 90
## Variables: 10
## $ hightemp   <int> 83, 73, 74, 95, 44, 69, 66, 66, 80, 7
## $ lowtemp    <int> 50, 49, 52, 61, 52, 54, 39, 38, 55, 4
## $ avgtemp    <dbl> 66.5, 61.0, 63.0, 78.0, 48.0, 61.5, 5
## $ spring     <int> 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1
## $ summer     <int> 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0
## $ fall       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
## $ cloudcover <dbl> 7.6, 6.3, 7.5, 2.6, 10.0, 6.6, 2.4, 0
## $ precip     <dbl> 0.00, 0.29, 0.32, 0.00, 0.14, 0.02, 0
## $ volume     <int> 501, 419, 397, 385, 200, 375, 417, 62
## $ weekday    <fctr> 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1,
```

Simple linear regression

In a simple linear regression model, there is a *single quantitative explanatory variable*. It seems reasonable that the high temperature for the day (`hightemp`, measured in degrees Fahrenheit) might be related to ridership (`volume`), so we will explore that first.

```
RailTrail %>% ggplot(aes(x = hightemp, y = volume)) +
  geom_point()
```



A simple linear regression model is of the form

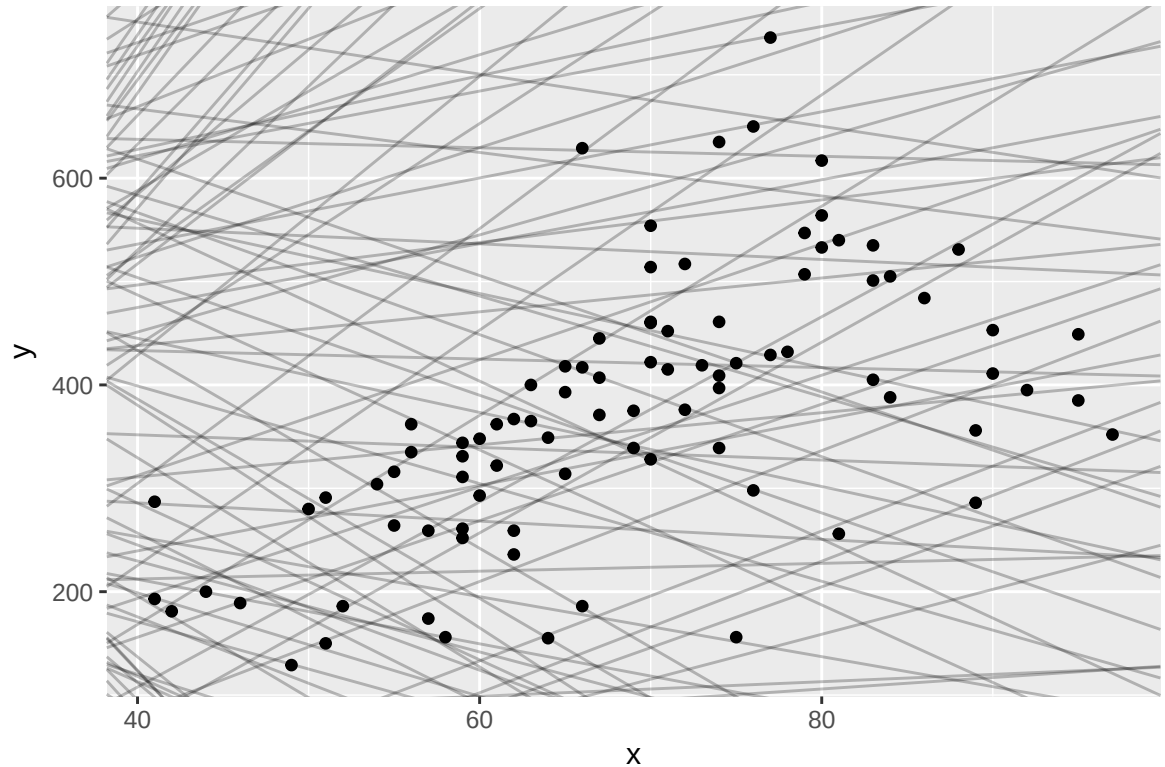
$$\text{volume}_i = \beta_0 + \beta_1 \text{hightemp}_i + \epsilon_i,$$

where $\beta_0, \beta_1 \in (-\infty, \infty)$ are the (unknown) *parameters* of the model. This is not just one model, rather *a family of models*, where different values of β_0, β_1, i.e. *parameters*, indicate different members in the family.

Here, ϵ_i stands for the unexplained error associated with the ith case. Often, the errors are assume to have mean 0 and variance σ^2. The model has a nice interpretation.

- If hightemp is 60, then volume is centered at $\beta_0 + \beta_1 \text{hightemp}_i$ with margin of error σ.
- If hightemp is increased by 1, then volume is increased by β_1 on average.

Fitting a regression model amounts to find *estimates* of β_0, β_1 (or to find a suitable member in the family). Delve into this idea by sampling some members of the family.



Each line segment is given by the formula $y = b_0 + b_1 x$, and is a member of the linear regression model family. A lot of these do not follow the apparent trend in the scatter plot. How do we choose a "good fit"?

There are million different ways to fit a model to data, but a standard technique is called *least-squares*.

Here is the idea behind the least-squares fit. We choose two of the models in the family:

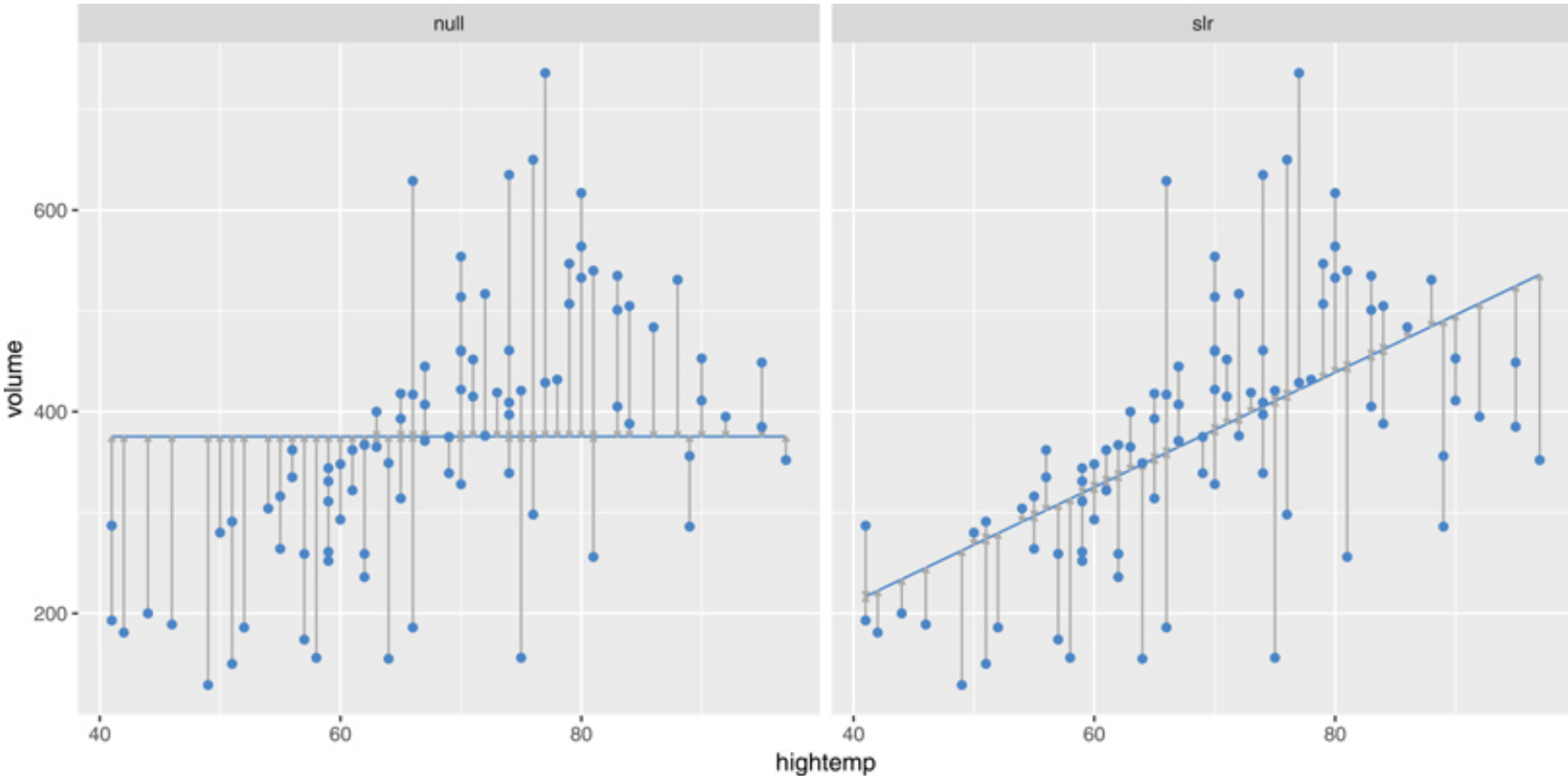


Figure 1:

Compare the errors of the fit on the left and the right:

```
##    x   y error.left error.right
## 1 41 193 -124.16667   -23.69772
## 2 41 287 -218.16667    70.30228
## 3 42 181 -112.16667   -41.39960
## 4 44 200 -131.16667   -33.80336
## 5 46 189 -120.16667   -56.20712
## 6 49 129  -60.16667  -133.31275
```

The *sum of squares of errors of a model* is

```
mod.compare %>%
  summarize(
    SSE.left = sum(error.left^2),
    SSE.right = sum(error.right^2)
    )
```

```
##   SSE.left SSE.right
## 1  9904439  955214.1
```

Of the five hundred models shown above, you can repeat this process of computing the SSE, and look for the model with the **least** sum of **square**d of errors. That is the best model among your collection of 500 models.

If you do this for all possible models (not just for the handful of 500 models), and find the "least-square" model, that is the best model fitted to your data. Mathematically, this is an optimization problem:

Find $\hat{\beta}_0, \hat{\beta}_1$ *such that*

$$\sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2 \leq \sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$

for all β_0, β_1 *in the family.*

Mathematics sometimes gives you an analytic solution: The formula you have learned in introductory statistics for $\hat{\beta}_0, \hat{\beta}_1$ is the solution of the above optimization problem.

In R, `lm()` computes $\hat{\beta}_0, \hat{\beta}_1$ (and a lot more):

```
mod <- lm(y ~ x, data = RTsim)
coef(mod)
```

```
## (Intercept)           x
##  -17.079281    5.701878
```

Many sophisticted statistical methods or "machine learning" algorithms are no unlike the simple linear regression. Essentially,

> *any statistic method optimizes a goodness-of-fit measure over a family of models.*

- Sometimes, the optimization comes down to a formula.
- Oftentimes, algorithms are used to numerically approximate the optimal member of the family. This is where efficient computation comes in.
- More often than not, you will want to consider *multiple* families of models. Computational efficiency becomes more important.

Quantifying uncertainty in regression

Thus far, we have fit a model and interpreted its estimated coefficients.

Take one step back. We have fit a model using *the data set we have*. More often than not, a data set is merely a *sample* from a *population.* What can we say about the population from our fitted model?

Uncertainty quantification arises in two ways:

1. In prediction of "y" from "x"
2. In estimation of the coefficient, i.e., β_1.

These are easy to answer if you *know* the population.

If you do not know the population, it is still possible, using *bootstrap*.

Case that you know the population

You never know about the population. Instead you make an *assumption* that you know certain things about the population.

Make the assumption that the members of the population following the rule:

$$y = \beta_0 + \beta_1 x + \epsilon_i, \epsilon_i \sim N(0, \sigma^2).$$

Even though you do not know the exact values of the parameters $(\beta_0, \beta_1, \sigma^2)$ of this rule/*model*, this is a big assumption!

(It is also typical to make assumptions on the sampling procedure, e.g. each member in the sample is picked independent from other members.)

Uncertainty in estimation of the coefficient

Use our understanding of the normal distribution to make inferences about *the true value of regression coefficients*.

- Test a hypothesis about β_1 (most commonly that it is equal to zero) and find a confidence interval (range of plausible values) for it.

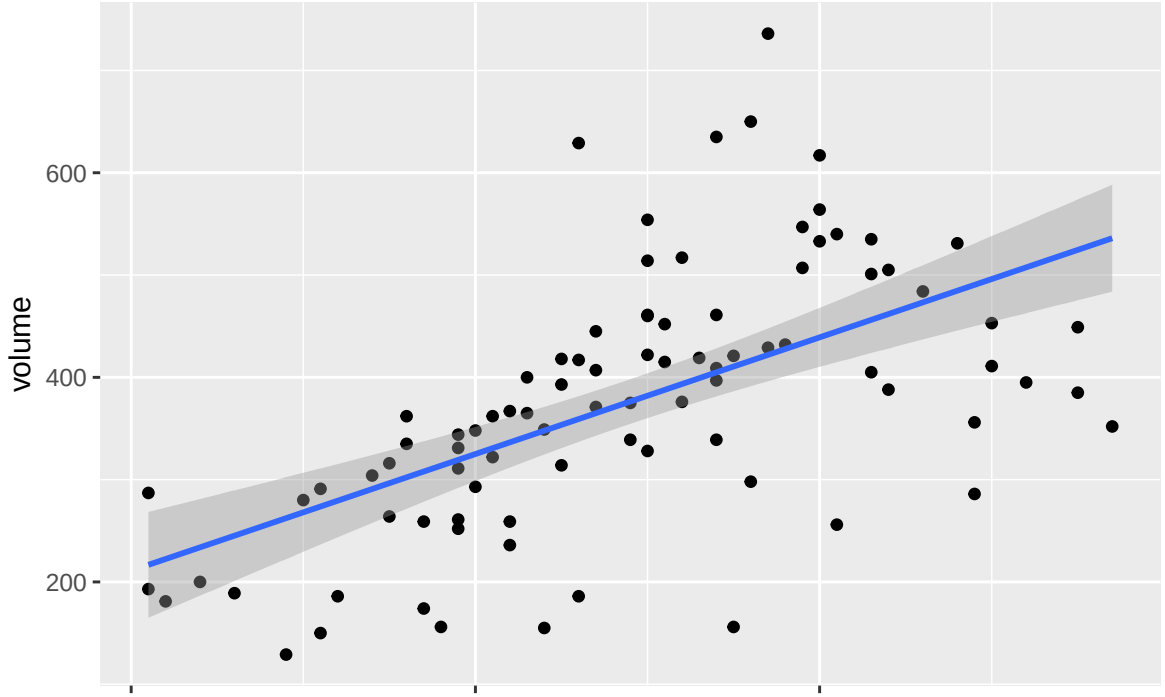
```
mod1 <- lm(volume ~ hightemp, data = RailTrail)
summary(mod1)
```

```
##
## Call:
## lm(formula = volume ~ hightemp, data = RailTrail)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -254.562  -57.800    8.737   57.352  314.035
##
## Coefficients:
```

Uncertainty in prediction of "y"

Likewise, the margin or error in the prediction can be found, for each value of "x", explaining that the true value lies in the (vertical) interval with 95% of confidence.

```
RailTrail %>%
  ggplot(aes(x=hightemp, y=volume)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE)
```



Case that you do not know the population

The previous case (that is, you ~~know~~ assume that population is *normally* distributed) is in fact very common. In case you do not want to make such an assumption, bootstrap gives a computationally heavy, but *distribution-free* answer.

This can be easily done in R for a simple model like linear regression.

- Implement bootstrap using R packages `dplyr` and `broom`.

We compute the bootstrap standard error for the estimate of β_1 (the coefficient) below.

```
library(broom)
boot <- RailTrail %>%
  bootstrap(100) %>%
  do(tidy(lm(volume ~ hightemp, .)))

head(boot)
```

```
## # A tibble: 6 x 6
## # Groups:   replicate [3]
##   replicate        term   estimate  std.error  statistic
##       <int>       <chr>      <dbl>      <dbl>      <dbl>
## 1         1 (Intercept)  -6.889481 64.0207424 -0.1076133
## 2         1    hightemp   5.474549  0.9210045  5.9441070
## 3         2 (Intercept) -69.060686 59.4112888 -1.1624169
## 4         2    hightemp   6.719702  0.8612593  7.8021819
## 5         3 (Intercept) -80.979445 59.6455792 -1.3576772
## 6         3    hightemp   6.827259  0.8556984  7.9785811
```

```
boot %>%  ungroup() %>%
  filter(term == "hightemp") %>%
  summarize(se = sd(estimate, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##          se
##       <dbl>
## 1 0.8190253
```

The bootstrap standard error is indeed quite close to the standard error estimated assuming the normality.

Categorical explanatory variables

Suppose that instead of using temperature as our explanatory variable for ridership on the rail trail, we only considered whether it was a weekday or not a weekday (e.g., weekend or holiday). The indicator variable `weekday` is binary (or dichotomous) in that it only takes on the values 0 and 1. (Such variables are sometimes called *indicator* variables: `weekday = 1` if weekday, 0 if weekend.) This new linear regression model has the form:

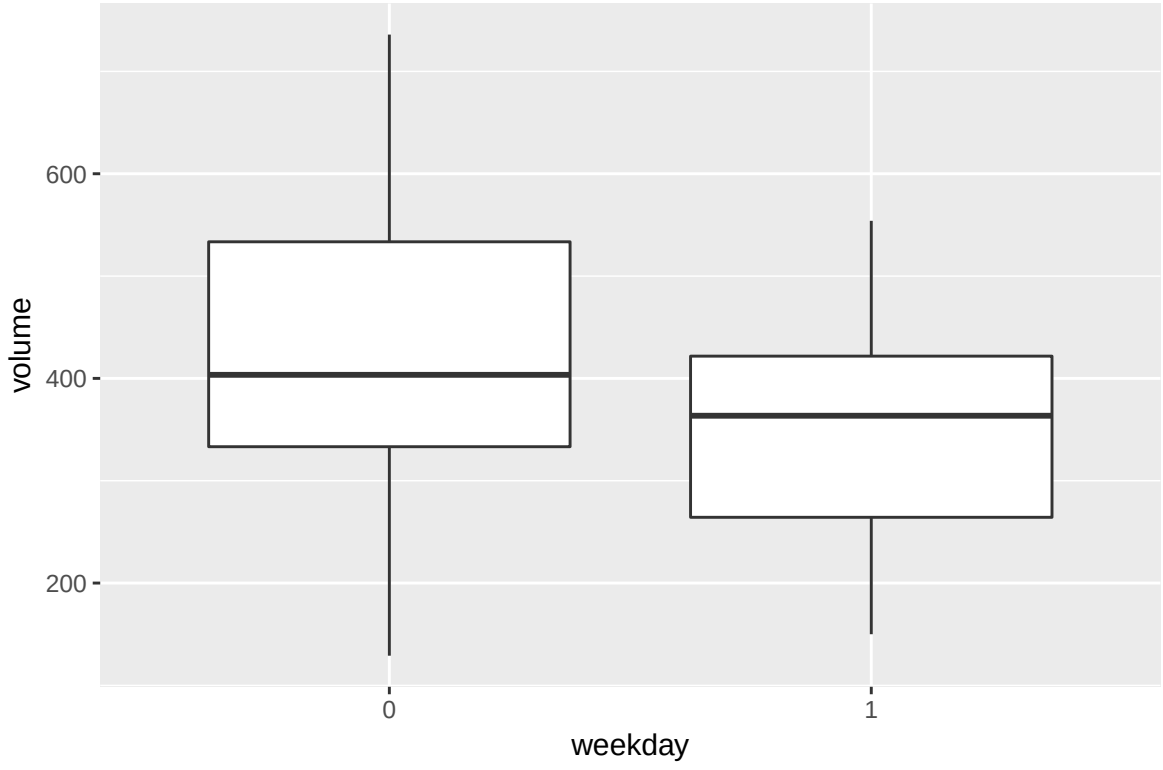
$$\text{volume}_i = \beta_0 + \beta_1 \text{weekday}_i + \epsilon_i.$$

There are now only two cases in the model: `weekday = 1` or 0.

```
RailTrail %>%
  group_by(weekday) %>%
  summarise(mean_volume = mean(volume))
```

```
## # A tibble: 2 x 2
##   weekday mean_volume
##    <fctr>       <dbl>
## 1       0    430.7143
## 2       1    350.4194
```

```
RailTrail %>%
  ggplot(aes(x = weekday, y= volume)) +
  geom_boxplot()
```



Is `weekday` related to `volumn`?

The least-squares fit does reflect our finding:

```
coef(lm(volume ~ weekday, data = RailTrail))
```

```
## (Intercept)    weekday1
##   430.71429   -80.29493
```

By default, the `lm()` function will pick the alphabetically lowest value of the categorical predictor as the reference group and create indicators for the other levels (in this case "0"). As a result the intercept is now the predicted number of trail crossings on a weekend. The interpretation of the model is that on a weekday, 80 fewer riders are expected than on a weekend or holiday.

Multiple regression

Multiple regression is a natural extension of simple linear regression that incorporates multiple explanatory (or predictor) variables. It has the general form:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_p x_{pi} + \epsilon_i.$$

The estimated coefficients are now interpreted as "conditional on" the other variables???each β_j reflects the predicted change in y associated with a one-unit increase in x_{ji}, conditional upon the rest of the explanatory variables. This type of model can help to disentangle more complex relationships between three or more variables.

Examples

Once you decide which variables to include in the model, R function `lm()` gives the least-squares fit for β_j:

```
mod1 <- lm(volume ~ hightemp + weekday, data = RailTrail)
mod2 <- lm(volume ~ hightemp + cloudcover + weekday, data =
mod3 <- lm(volume ~ . , data = RailTrail)
```

Here, the model formula `volume ~ .` is the *full* model, i.e. a model with all available explanatory variables in the data set.

We now encounter another layer of complexity (First layer was the fitting itself; oftentimes fitting a model is merely an educated trial-and-error procedure.):

> *Among the three families of models `mod1, mod2, mod3`, which model family should we use?*

Variable selection problem

In multiple linear regression, this question is translated to a *variable selection problem*: Among the p variables, which variables to use?

- When only a few variables are included, the model may not well explain the pattern in the data (*underfit*).
- When too many variables are included, the model may be swayed by idiosyncratic errors and peculiar values (*overfit*).
  - In this case, there may be problem associated with the fitting procedure (in this case, the least-squares). In fact, some coefficients of `mod3` are not estimated. This is caused by a problem called *colinearity*. To cope with this, one may choose to alter the fitting procedure (e.g. regularized or sparse regression).

Computationally expensive data-driven modeling

The degrees of freedom in the choice of models are now almost infinite. Even sticking with the multiple linear regression, you will need to decide

1. which goodness-of-fit measure to minimize?
2. over which family of models?

Can't decide? We can use the data (just like we did for bootstrap) to make informed decisions on these two questions. The way we seek for answers is again an educated trial-and-error procedure, which tends to be very computationally expensive. Statisticians are now required to take both inferential and computational aspects of model fitting. This is modern statistics. Some other people call it data science.

We will see some basics of data-driven model fitting later.

- Note: Understanding multiple linear regression is so fundamental in understanding any other methods. This lecture is a short, insufficient introduction of regression.

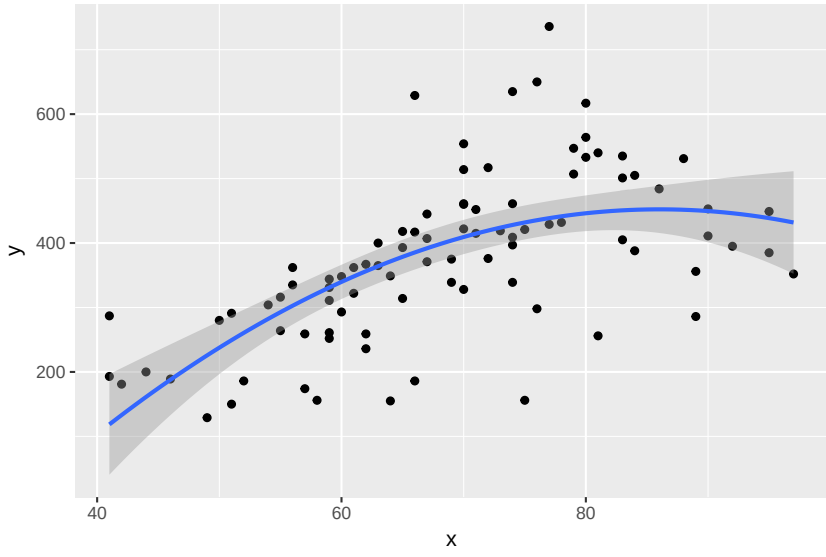
Nonparametric regression a.k.a. scatterplot smoothing

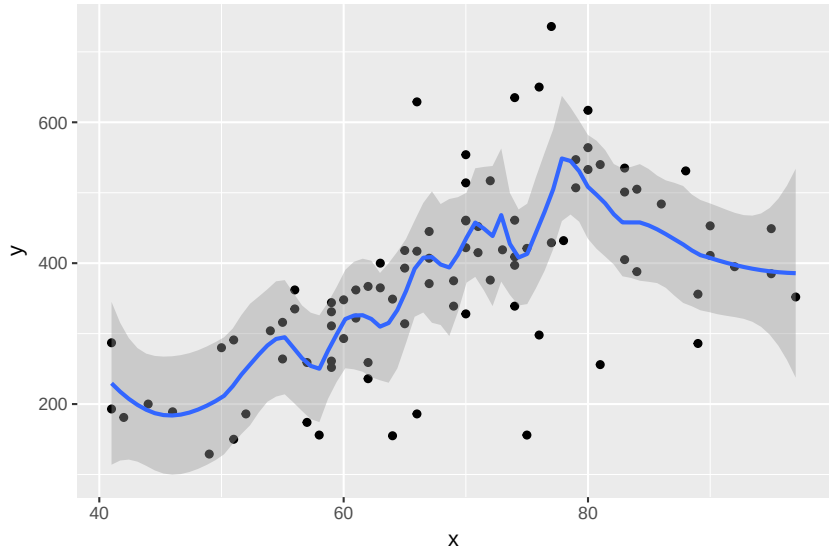
Nonparametric regression assumes a simple (not necessarily linear) model:

$$y_i = f(x_i) + \epsilon_i.$$

where f is a *smooth* function. The family of models is parametrized by the function f, which inself is parameterized by a near infinite number of parameters. "Nonparametric" regression is named so because there are the infinite number of parameters to fit. As a result, the fit is very flexible.

```
## `geom_smooth()` using method = 'loess'
## `geom_smooth()` using method = 'loess'
```





```
scat_ex <- RTsim %>% ggplot(aes(x,y)) + geom_point()
scat_ex + geom_smooth(span = 10)
scat_ex + geom_smooth(span = 1/5)
```

In the above example, `ggplot2` automatically use a local polynomial regression implemented in `loess()`. There are other well-developed methods, such as kernel regression implemented in `ksmooth()`. The choice of fitting methods comes from the choice of model families.

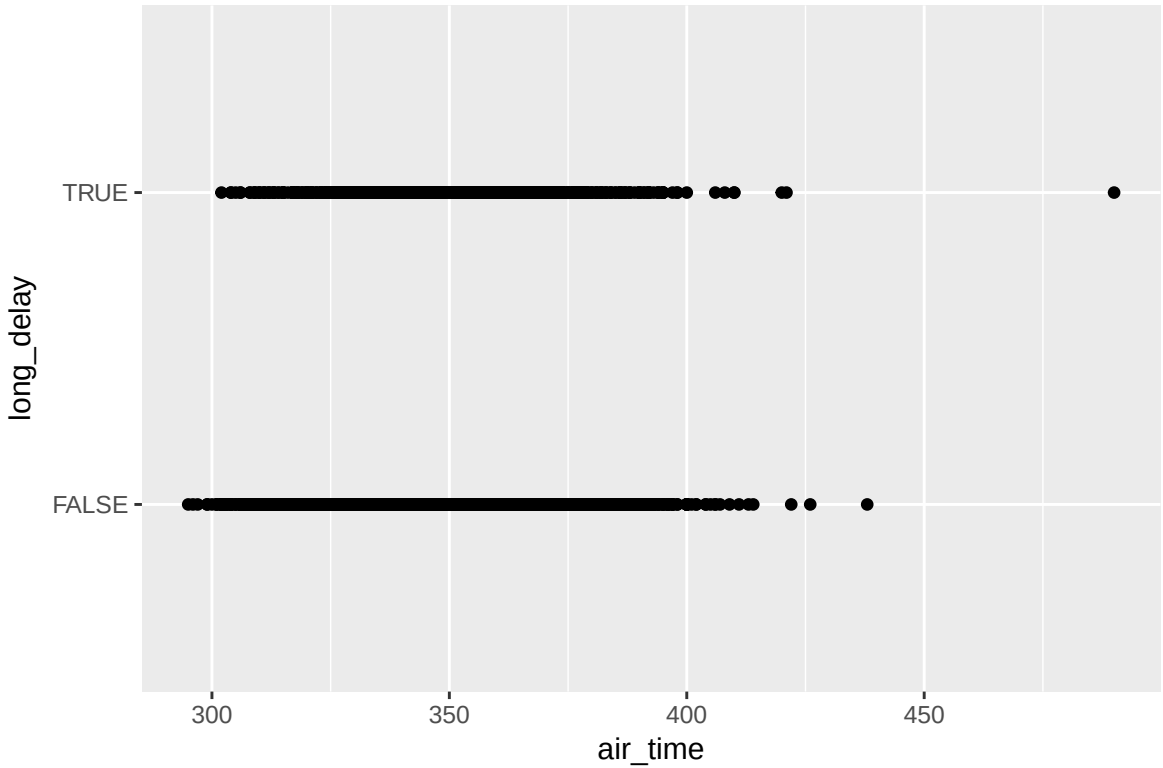
Notice that for two different values of `span`, the fit looks quite different. The `span` is an example of tuning parameter. Data-driven choice for the value of the tunining parameter is well-studied and we will glimpse over it later.

Categorical response variable

Our previous examples had quantitative (or continuous) outcomes. What happens when we are interested in modeling a dichotomous outcome? For example, we might model the probability of long delay as a function of airtime (We are back to the "SF" example).

It would be very awkward to use a linear model to fit the following relationship.

```
# SF_longdelay <- SF %>% mutate(long_delay = arr_delay > 60
SF_longdelay %>% ggplot(aes(x = air_time, y = long_delay))
  geom_point()
```



The main difficulty is caused by the fact that the response has only two possible values $\{0, 1\}$ as opposed to $(-\infty, \infty)$ in the simple linear regression. *Logistic regression* can be used in this case, which is an example of generalized linear regression. Generalized linear regression transforms the y-coordinate (and a lot more) so that a highly interpretable linear model can be used.