Table of Contents

Introduction

Knock Software's *Ride Report* app combines a simple thumbs-up/thumbs-down rating system with GPS traces of bicycle rides to compile a crowdsourced data set of which routes are and are not stressful for urban bicyclists.

From the user's perspective, the app that collects the data is simple: *Ride Report* automatically detects when a user start riding their bike, records the GPS trace of the route, and then prompts the user at the end of the ride to give either a thumbs-up or thumbs-down rating. From this, they were able to create a simple "stress map" of Portland, OR, which displays the average ride rating of rides going through each discretized ride segment.

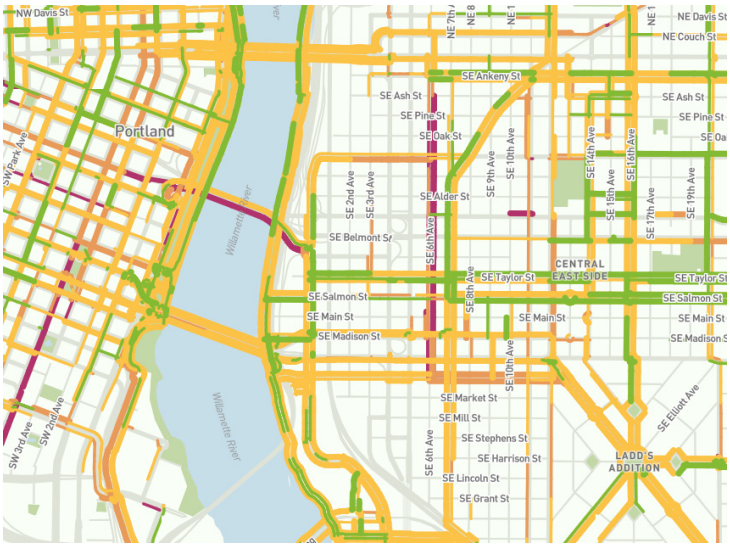

Figure 1: Ride Report's Stress Map for Portland. Greener road segments incidate less while more red segments indicate more stressful streets. "Stress" computed by taking the average rating for each segment.

The app privileges reducing barriers to response to increase sample size over ensuring quality and consistent responses. This presents the first problem: how can we analyze ratings when riders are likely rating rides inconsistently?

At the same time, we have another challenge. We have ratings associated with routes, but we would like to know the effect of particular road segments, for both inference (what effect does this road segment have on the rating?) and prediction (given a route, what do we expect the rating to be?) purposes.

Finally, there are interesting issues with missing data. First, the sample of rides we have are biased towards routes that riders percieve as better. Second, a significant portion of the rides are missing a response, and non response is unlikely to be independent of the response. The good news is that we have all the predictors for every ride, enabling us to build a model that is able to leverage the data with missing responses rather than listing the missing data as a liability.

0.1 Accounting for Rider Rating Variance

For ratings we are interested in modeling variance between riders (as we might expect riders to have different rating patterns than their peers) and within riders (as riders may not rate the same route and conditions the same every time). To model this, we propose using multilevel regression, with random effects from each rider. This approach has been used in similar situations, in one case to model sexual attraction[1].

In a multilevel model, we fit a regression where the intercept term varies by group but comes from a common distribution. For example, if we let r_i be the rating of the ith ride, X_i be the ride-level variables, then we can fit a regression:

$$\mathbb{P}(r_i = 1) = \text{logit}^{-1}\left(\alpha_{j[i]} + X_i\beta\right),$$

where α_j is an intercept specific to rider j. In addition, the rider intercepts come from a common distribution,

$$\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2),$$

where μ_α is the mean of all the α_js.

Using varying intercepts will allow our model to exhibit the property of varying rates of riders giving stressful ratings. This can be extended to model differences in how riders respond to different kinds of road conditions using varying slope models. We explore multilevel model further in Section Section 2.2 and multilevel models for riders in Section Section 3.1.

0.2 Addressing Road Segments as a Level

We examine multiple approaches to modeling road segments. In the first, we regard add regression term for the route, that is a weighted sum of terms for each road segment in the route, weighted by the lengths of the segments. The terms for the

the contribution of the route to be a weighted sum of the contributions of road segments in the route, weighted by the lengths of the segments.

0.3 Approaching Missing Data at Multiple Levels

This data set contains missing data at two levels.

[1] Mackaronis et al. (2013)

First, there are many routes without any ratings. The routes taken by riders are not a random sample of routes, but are often already chosen by the rider as the least stressful ride. As we might expect because of this bias, only a small proportion of rides are rated as "stressful."

One solution we explore to this problem is adding a segment popularity parameter as a segment-level predictor.

Second, not every ride is given a rating. We do have the route they chose, and all associated covariates, but the response variable, rating is missing. As we will discuss in chapter Chapter 5, the pattern of non-response is unlikely to be unbiased.

To address the second problem, we first build a separate probability model that predicts non response, and then integrate that model into our main model.

Chapter 1

Data Sources

We combine several data sources to do our analysis. Information about individual rides, including the GPS trace, the rider, and start timestamp were provided by Ride Report. Weather data were collected from Weather Underground's archive of the KPDX weather station and a Portland Fire Bureau station.

Our goal in this chapter is to discuss this data and what considerations we should have in mind before exploring it in depth. This includes how and by whom the data was collected, who and what is this data actually representative of, and what samples were taken of the data.

Some of these considerations, such as the limited demographics represented in the Ride Report data, pose serious limitations to how our inferences can be generalized. Others, such the large number of missing responses in the Ride Report data, motivate the analysis we are doing in this thesis. Finally, there are other considerations which we will acknowledge here, but addressing them is out of the scope of this thesis. This data set contains an abunance of potential research questions, only a fraction of which could be reasonably addressed in one thesis.

1.1 Ride Report

Ride Report's data is the focus of this paper. Knock Software created the app to collect large amounts of information about urban cyclists' routes and experiences on those routes. The hope is that this information will be valuable to city planners.[1]

Ride Report's approach to crowdsourcing this data is particularly important to understand. The app automates every piece of the data collection process except for the rating given by the rider. Thus, the app casts aside nuanced and (somewhat) reliable human input in favor of increasing sample size: one could imagine a similar app where users have more control over how the route is recorded, have the ability to rate on a more fine-grained scale, and are given more direction in what they are rating for. This tradeoff causes two problems with the reliability with the data.

Before we get into the potential issues in the data collection, though, let's examine

[1] Knock's other project is making a cheaper bicycle counter for cities to monitor traffic flow, again intended to be sold to cities wishing to improve bike infrastructure.

the data collection process itself. When installed on a persons phone, the Ride Report app attempts to automatically detect when the user starts riding their bicycle, based acceleromoter data, when a user leaves a familiar Wi-Fi network, and some other pieces of information. When the app detects the start of a ride, it starts recording a GPS trace. At the end of the users ride, the app detects them getting of their bike (in a similar process to how it detected the start of a ride) and prompts them to give a rating of the ride. The ride data is saved then, even if the user does not provide a rating.

This automatic detection of when a ride starts and stops leads to two related and common errors in the dataset: first, one ride is often split into two or more rides at points, such as at a stoplight or a train crossing, where a cyclist stops for an extended period of time; second, car rides are sometimes misclassified as bicycle rides and vice-versa (car rides are not rated.) The app allows riders to correct the misclassification, but provides no way to join split rides back together.

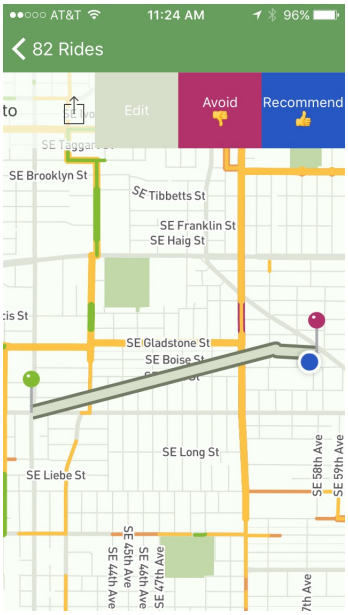

Figure 1.1: The Ride Report app's interface has changed significantly between versions, including the rating text displayed after a ride. This is the current version as of Februrary 2015.

The app only recently became publicly availible and has undergone significant changes in the course of it's life. In particular, while the ratings have always been binary, the labels have changed at various points in time. For a while the rating labels were "Stressful" and "Chill", while now they are labelled "Recommend" and "Avoid" (see Figure 1.1.) Other fundamentals of the data collection process, have remained constant.

The data colletion method itself has some problems, but there also may be some biases in the population of riders using the app. The app is only availible on iOS, so only iPhone owners could use this application which may imply a bias toward riders of higher socioeconomic status. At the time of the start of the thesis, the app was in

private beta, meaning only people who actively sought out using the app were able to use it. Now the application is public and on the Apple App Store, making it more widely availible. So many of the earlier rides may be people within the developer's personal network. Unfortunately, it's hard to make any solid conclusions about the users of the app because Ride Report doesn't collect any demographic data about their riders.

One other issue with the Ride Report data guided our analysis: privacy. Because the data involves timestamps and GPS locations of people's commutes, the data is very sensitive: one could easily infer someone's home and workplace based on their most commons routes. In fact, this data is protected by an end-user license agreement (EULA) which prevents sharing of data, without the explicit permission of those involved. This presented a logistic challenge: how were we to do inference and data exploration without access to the data?

By agreement with Knock Software, identifying data must be kept private. With permission from five riders, Knock was able to give us a small subset consisting of all the rides from those five riders, to be kept confidential. That is the data set we used for prototyping models and some basic exploratory analysis. Knock also agreed to allow us to run models fitting scripts on larger samples of their dataset, as long as they were performed on their computers, with no identifying data leaving their system.

While at first this set up seems like an inconvenience, it actually has some advantages. One of the pitfalls of having an entire data set, especially a high dimensional one, is that in performing exploratory analyses it is often too easy to find spurious "statistically significant" results. Instead, we must come up with our models before running them, greatly limiting the choices we can make in the garden of forking paths.

1.2 Weather Data

Slippery roads and formidable winds are no fun for anyone balancing on a two-wheeled vehicle. Weather is, then, one of the most obvious family of predictors for ride rating, at least intuitively. We use the time of a ride to join in data about the weather conditions during the ride, including

- the temperature,
- whether and how much it is raining,
- whether the roads are wet or have puddles,
- wind and gust speed.

We include the first two, temperature and percipitation, to account for rider comfort. A swealtering, frigid, or stormy day could make an unpleasant experience for a bicyclist and thus could lead to more negative ratings.

On the other hand, we include the last two, wet road and gust speed, as factors that impact safety. During and after storms, puddles often accumulate in bike lanes before the center of the road, pushing cyclists into lanes shared by cars, which are often more dangerous.

Gust speeds impact the aerodynamics of a ride, which are particular important for bicyclists. It's one of the main reasons cyclists care about getting into lower (and more aerodynamic) rider positions. Thus, high wind or gust speeds may affect rider rating.

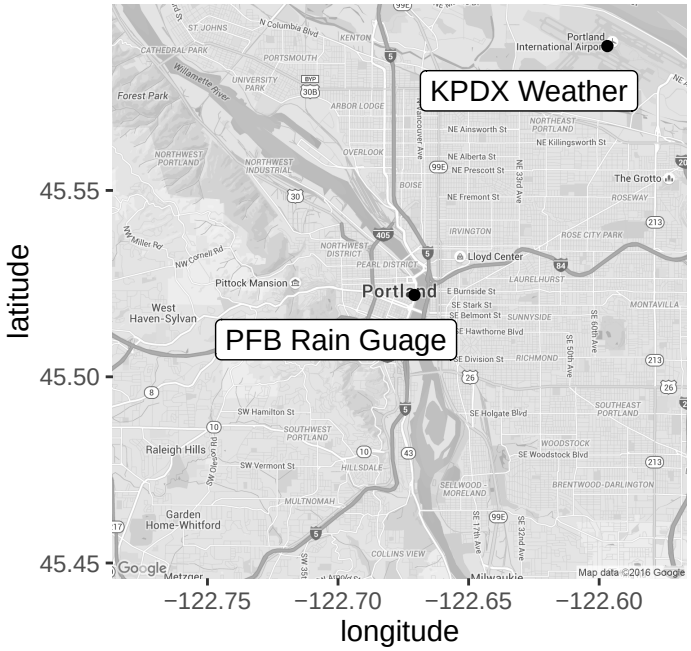

Figure 1.2: Positions of weather data collection sites. Daily weather information was collected at the KPDX weather station at Portland International Airport. Hourly percipitation data was collected at the Portland Fire Bureau's rain guage in downtown Portland.

We are limiting our study to rides in Portland, Oregon. Given this, we can first assume that it may be reasonable to expect that riders are used to the same climate, and thus have somewhat similar responses to weather. This also makes it reasonable to use data from one nearby weather station, rather than attempting to collect from several stations and creating a spatial model for weather.

For daily summaries of weather conditions, we used weather history from the KPDX weather station at Portland International Airport downloaded from Weather Underground. From this we were able to get daily weather data, including

- Average, minimum, maximum temperature for the day.
- Total percipitation.
- Mean wind speed, as well as gust speed (speed of brief, strong winds.)

We also got hourly rainfall data from a data stream at the Portland Fire Bureau Rain Gage at 55 SW Ash St., which is just about the geographic center of Portland. This just gives raw uncorrected rain guage data, but gives us a fine grain look at how much rain there has been recently.

For daily weather data, such as temperature highs and average wind speed, we use information from the KPDX weather station. This weather station is the best calibrated in the area. It is further from the geographic center of the rides we are examining, but because the weather is daily summary statistics, we don't expect closer weather stations to be much more informative. Figure 1.2 shows the geographic positions of these two stations.

1.3 Notation for the Joined Data Set

We combined the ride records with weather data by joining by start datestamp of the ride. We will denote our set of ride-level predictors, each of which is a column vector as, - x^{length}, ride length, scaled to have mean 0 and standard deviation of 1 - x^{rain}, rainfall during hour of ride, in tenths of inches - x^{rain4h}, rainfall during past four hours before ride, in tenths of inches - x^{wind}, mean wind speed for the day, in miles per hour - x^{gust}, max gust speed for the day, in miles per hour[2] - x^{temp}, average temperature, in degrees Fahrenheit for the day.

We will often represent this set of predictors as the ride-level predictor matrix $X = (x^{\text{length}}\ x^{\text{rain}}\ x^{\text{rain4h}}\ x^{\text{wind}}\ x^{\text{gust}}\ x^{\text{temp}})$. We also have the predictor $t \in [0, 24]^n$, representing the time of day of the ride, measured by hours since midnight. (Because of it's special role in building our models, we use the simple notation of a single letter for it.)

Let $Y_i = 1$ if the ith ride received a negative rating, and $Y_i = 0$ if it received a positive rating. Choice of coding which events are 0's and which are 1's is arbitrary when making logistic regression models, though we made this choice because for the sake of our analysis, negatively rated rides are more interesting events. For urban planning applications, they define the areas that need attention.

[2]Gust speeds are the max speeds of winds that are fast, highly variable, and short-term. For METAR weather stations, which the KPDX station is, gust speeds report the maximum wind speed when there were rapid fluxations in wind speed with at least 10 knots in the difference between the lows and highs.

Chapter 2

Methods

We use many statistical methodologies in this paper. We outline here the central methods used, both to familiarize the reader and to establish the notation we use throughout this paper. The models we present combine logistic regression, multilevel[1] models, additive models, and smoothing splines. We also make use of two recently developed graphical model evaluation tools: the separation plot and the heat map plot.

The one large methology not covered here is the expectation maximization algorithm we use to model the missing data mechanism for missing ride ratings. The theory of that algorithm is outlined in Chapter 5 and an example of its implementation can be found in Appendix A.

2.1 Logistic Regression

Statistical models are often split into regression models—models with a quantitative response—and classification models—models with a categorical response. Thus, it may seem odd that we are using a regression model when our response variable, ride rating, is a binary outcome.

But, when modeling a binary variable Y, we consider it a Bernoulli random variable,

$$Y = \text{Bernoulli}(p),$$

where p is the probability the response is 1 and $1 - p$ is the probability the response is 0. So our response variable Y may be binary, but the primary quantity of interest behind that outcome is p, a quantitative variable. This is why we consider logistic regression a regression method.

Logistic regression is one form of generalized linear regression. Recall that in linear regression we use data with response variable y_i and j predictors $x_{i1}, \ldots, x_{ij}$ to fit the best-fitting linear function

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_j x_{ij} + \epsilon,$$

[1] Multilevel models are often referred to as hierachical models or mixed effects models.

where $\epsilon \sim N(0, \sigma^2)$, by estimating $\beta_0, \ldots, \beta_j$. We can equivalently write,

$$y_i \sim N(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_j x_{ij}, \sigma^2).$$

We could, in fact, try to predict p with a linear regression, though such a model will always have the problem of predicting probabilities outside of the range of $[0, 1]$. That's not a recipe for simple interpretation or reliable predictions. Generalized linear regression uses a "link function," g, to modify the regression so the range of the response more accurately reflects the practical range of the variable; i.e. model:

$$g(y_i) = \beta_0 + \beta_1 x_1 + \ldots + \beta_j x_j + \epsilon.$$

In logistic regression, the "link" function is the logit function, $\text{logit} : [0, 1] \to \mathbb{R}$,

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right).$$

This function is also known as the log-odds, because odds are defined as $p/1 - p$ for any probability p.

So in logistic regression, we model the probability of $y_i = 1$ as,

$$\mathbb{P}(y_i = 1) = \text{logit}^{-1}(\beta_0 + \beta_1 x_1 + \ldots + \beta_j x_j).$$

Notice that the inverse logit function[2] maps values from $\mathbb{R}$ to $[0, 1]$. Thus, the function provides a convenient way to map linear combinations of other variables onto values that are valid probabilities. Other such functions exist and are also used for regressions with binary responses, such as the probit function. Logistic regression, however, is easier to interpret—because of the odds connection—and more efficient to compute.

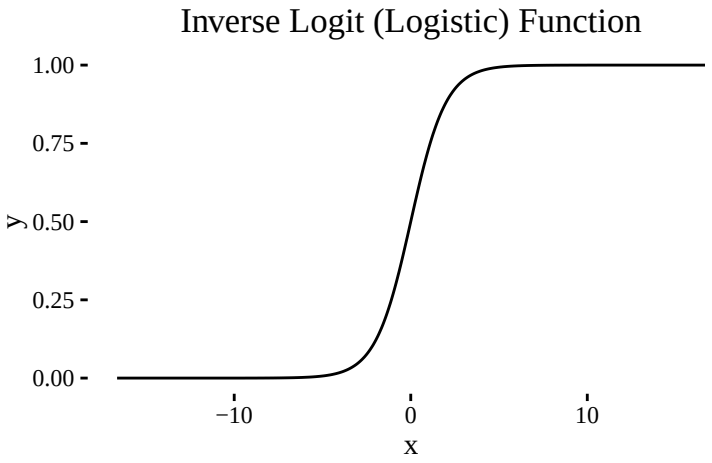

Figure 2.1: The inverse logit function gives a convenient way to map linear combinations of real numbers to valid probability values.

[2] sometimes called the logistic function

Though coefficients from a logistic regression can't be interpreted the same way as a linear model, they do have a convenient interpretation. Because the linear part of the model represents log odds, the coefficients are log odds ratios; That is, the exponentiated coefficients $e^{\beta_1}, \ldots, e^{\beta_j}$ represent the multiplicative effect a one-unit increase in the corresponding predictor has on the odds. For example, if we fit a simple logistic regression with $\mathbb{P}(Y = 1) = \text{logit}^{-1}(\alpha + \beta + X)$, we could interpret $e^{\beta} = 1.1$ as meaning that a one unit increase in X gives a 10% increase in the odds that $Y = 1$.

We started out talking about the ways logistic regression is both a classification and regression model. We will mostly care about logistic regression as a regression model. One of the main ways we will assess our models is through the accuracy of predictions. But, in the classification framework, in order to predict the outcomes $\hat{Y}$, we need to choose a probability threshold to decide which outcome we should predict for a given observation. When we care mainly about prediction, this approach can make sense. Our interest, however, lies in prediction. So rather than choose an arbitrary threshold, we will evaluate the accuracy of our predicted probabilities themselves. (How exactly do we do this? See Section 2.4.)

2.2 Hierarchical Models and Mixed Effects Models

Data often contain hierarchies. For example, a set of students' test scores may contain the hierarchy of districts and schools those students attend. Or a set of soil samples may have been taken at several distinct sites, thus having a hierarchy of sites and samples. In the bike ride data we examine, there is the hierarchy of riders and rides.

We will talk about different "levels" of variables corresponding to places in this hierarchy. When we refer to "ride-level variables," we refer to variables that are specific to a ride, whereas we refer to "rider-level variables" as those specific to the rider, and thus also all the rides that rider takes. For example, we consider length a ride-level variable and total number of rides taken a rider-level variable.

We will also discuss road segment-level variables, which are variables that are specific to the road segments in the route of a ride (e.g. length, presence of bike lanes, etc). But there isn't a clear road segment-ride hierarchy: each ride contains multiple road segments and each road segment is contained by multiple rides. Thus, this isn't a case where multilevel modeling is applicable. (The ideas behind it, though, may be fruitfully adapted, as we discuss in Chapter 6)

Gelman describes two traditional ways of dealing with hierarchical data that multilevel models contrasts with: "complete pooling" and "no pooling."[3] In "complete pooling," we ignore the group-level variables, and give identical estimates for parameters for every group. In "no pooling," we do entirely separate regressions for each group. Multilevel models are a comprimise between these extremes ("partial pooling", as Gelman calls it) where all the data is considered in a single regression with some parameters shared between groups and some different between groups.

These multilevel models work for other forms of regression, but we will focus on logistic regression, as it is the method we use in this paper. We will be using notation

[3]Gelman and Hill (2006), p. 7

adapted from Gelman and Hill's description of multilevel models.[4] Consider a data set composed of

- n observations of a binary response variable y_i, $i \in 1, \ldots, n$,
- p observation-level predictors $X_i = x_i^1, \ldots, x_i^p$,
- j groups in which the observations are split into,
- l group-level predictors $U_{j[i]} = u_{j[i]}^1, \ldots, u_{j[i]}^l$, where $j[i]$ is the group of the ith observation.

We could fit a model where the intercept varies by group:

$$y_i \sim \text{Bernoulli}\left(\text{logit}^{-1}\left(\alpha_{j[i]} + X_i\beta\right)\right), \tag{2.1}$$

$$\alpha_{j[i]} \sim N(\gamma_0 + U_{j[i]}\gamma, \sigma_\alpha^2), \tag{2.2}$$

where $\alpha_{j[i]}$ is the intercept for the jth group, β is a vector coefficients for the observation-level predictors, γ_0 are the group-level intercepts, and γ is a vector of coefficients for the group-level predictors. We could also specify a similar model where there are no group-level predictors, such that we simply have different intercepts for each group,

$$\alpha_{j[i]} \sim N(\gamma_0, \sigma_\alpha^2), \tag{2.3}$$

We can also consider a model that has slopes varying by group. For simplicity, let's consider just one observation-level predictor, x_i, that will have varying slopes $\beta_{j[i]}$ as well as one group-level predictor, u_j. We could specify the model as,

$$y_i \sim \text{Bernoulli}\left(\text{logit}^{-1}(\alpha_{j[i]} + \beta_{j[i]}x_i)\right), \tag{2.4}$$

$$\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} = N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha u_j \\ \gamma_0^\beta + \gamma_1^\beta u_j \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix}\right). \tag{2.5}$$

These models can be fit with maximum likelihood estimation using the `lme4` package in *R* or can be fit with Bayesian MCMC using *Stan*. The latter has the advantage of making it easy to estimate group-level uncertainty at the expense of more computation. We fit models using `lme4`, but make use of *Stan* when we have ride-level parameters we want to estimate, in Section 4.2.

2.3 Additive Models and Smoothing Splines

We want to explore using non-parametric methods to model the relationship with time and length, but we wish to keep the other parts of our model. We can do this with an additive model. Additive models assume that the response is the sum of functions of each of the predictors:

[4] Gelman and Hill (2006), p. 251–252

$$\text{logit}(\mathbb{P}(y_i = 1)) = \alpha + \sum_{j=1}^{p} f_j(x_{ij}).$$

These functions can be linear, so generalized linear regression is a subset of additive models. But more interestingly, these functions can be non-parametric.[5] One of the most common types of functions fit are smoothing splines.

Smoothing splines are essentially cubic functions stitched together at points called "knots" such that the full piece-wise function is continuous and has continuous first and second derivative. One can further define cyclic cubic splines, which simply have the constraint that the last knot and first knot are treated as the same, thus allowing a continuous cyclic function to be fit.[6]

Computation of multilevel additive models with splines is available in the `gamm4` package, which we use to fit the two following models.

2.4 Tools for evaluating models

After fitting our models, we will want to know, how do each of our models compare? Did adding a particular term enhance or diminish the accuracy of our model? Instead of focusing on one measure of fit we use several. Loglikelihood and AIC provide useful summaries of fits to the data based on the likelihood function. Separation plots—which we discuss in the next section—allow us to assess the predictive ability of a model; in particular, can the model identify high and low probability events? The Heatmap plot, another graphic tool we assess models with, show use how well the predicted probabilities fit the empirical probabilities of the corresponding observations; i.e. if the model says a set of observations will have $y_i = 1$ 60 percent of the time, is the percentage of those that have $y_i = 1$ equal to 60?

We also use area under the ROC curve (AUC) measure popular for assessing logistic regression. In particular, we compute 10-fold cross validated ROC statistics to detect if models are overfitting to the data.

2.4.1 The Separation Plot

The separation plot, created by Greenhill et. al.[7], is designed to show how well a logistic regression model can distinguish between high and low probability events.

Let y be a vector of observed binary response and $\hat{y}$ a vector of predicted probabilities of a 1 for each observation, predicted by some model. Then we can construct

[5] How are these models fit? Using what's known as the Backfitting Algorithm. We define the kth partial residuals $Y^{(k)} = Y - \left(\alpha + \sum_{j \neq k} f_j(x_j)\right)$. (That is, define the portion of Y leftover for $f_k(x_k)$ to fit to after the other f_j's have had their share.) Then, iteratively fit each of the functions f_j on the partial residuals $Y^{(j)}$ until each of the functions converge. For a further quick look at additive models, check out Cosma Shalizi's lecture notes (Shalizi (2013a))

[6] For a brief and entertaining introduction to smoothing splines, see Shalizi (2013b). For a more in-depth look at splines, check out Wood (2006)

[7] Greenhill, Ward, and Sacks (2011)

the plot as follows: We plot the data $(y, \hat{y})$ as a sequence of vertical stripes, colored according to observed outcome, y, and ordered from low to high probability based on $\hat{y}$. A curve is superimposed upon the stripes showing the $\hat{Y}$ as a line graph. And finally, a small triangle is placed indicated the point at which the two colors of lines would meet if all observations $y = 0$ were placed to the left of all the $y = 1$ observations; *i.e.* showing where the boundary would be if the two classes were perfectly separated by the model.

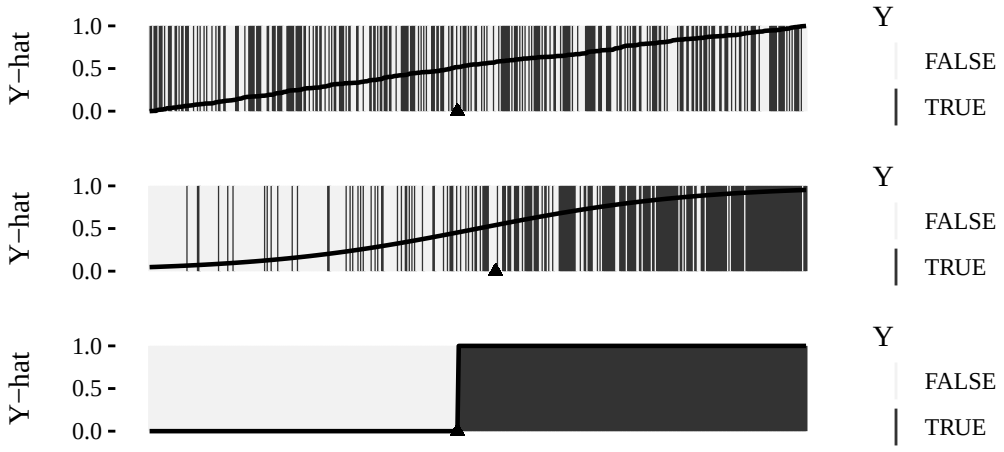

Figure 2.2: Above we have examples of three separation plots. The first plot shows what it looks like when Y and $\hat{Y}$ are uncorrelated. The second plot shows a fairly good model, where the Y are generated as Bernoulli($\hat{Y}$). The third plot shows a model where the responses are fully separated.

Separation plots don't do well with larger sample sizes: if there are two many observations, it becomes difficult to read. There are several ways around this, but we choose to randomly sample the observations.

2.4.2 Our Probability Plots

One graphical tool we adapted for this project maps probabilities of the outcome against continuous variables. We use this for exploration to understand the relationship between predictors and our binary response, but it's also useful for evaluating our models.

The goal of the plot is to show a measure of an emperical $\hat{p}$, the probability that the response is 1, on the y-axis, and a continuous variable X on the x-axis. A simple approach would be to bin the observations by the values of X, and within each bin, model the data as a bernoulli random variable and compute a local value for p.

The problem with the binning approach is that we get some error due to how we happen to bin things. An alternative is to do the same computation for a sliding window rather than a bin. This reduces the error due to our binning, but we still have to choose a proper window width. We will choose the window width that minimizes

the leave-one-out error (the sum of the squares of the error in predicting a single observations based on a regression fitted to the rest of the observations.)

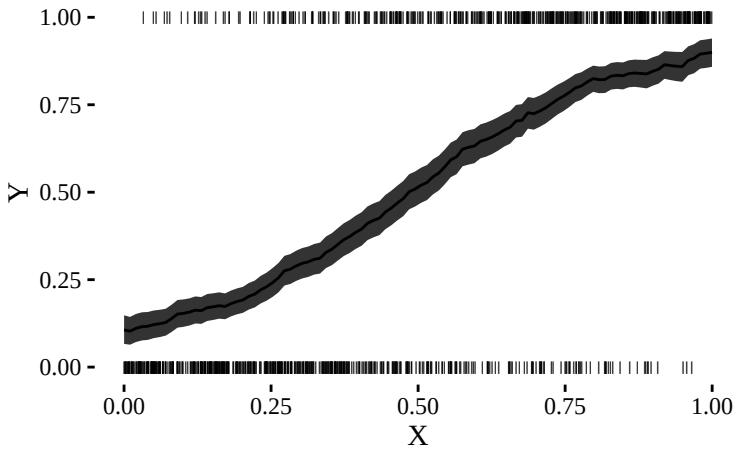

Figure 2.3: Example of probability plot. 1000 Xs were sampled from the distribution Uniform$(0, 1)$ and each Y_i was generated from the distribution $Y_i \sim \text{Bernoulli}(X_i)$. The smoothed curve shows an approximation of p for each value of X_i, with 90% confidence intervals in grey.

We also make use of this plot to compare the predicted probabilities for a testing set to the empirical probablilities in the data. J. Esarey and A. Pierce advocate using this in their heatmap plot to assess model fit, showing that such a plot will show model mispecification problems better than other measures commonly used, like AIC.[8]

[8]Esarey and Pierce (2012)

Chapter 3

Modeling Rides and Riders

Complex statistical models can accurately model intricate processes. But they also run the risk of overfitting to the data. To avoid this, we build up our models from simple to complex, comparing the models with cross validation to make sure the complexities introduced add real value.

In this chapter we focus on building models that incorporate information about rider, weather conditions, time of day, and ride length. In brief, our models start with a logistic regression model considering only ride-level variables, and formulate more complex models by adding various terms. Table 3.1 describes each model briefly along with the models label.

Table 3.1: List of models we evaluate in this chapter.

Model	**Description**
Model 1	(Baseline) logistic regression
Model 2	Add rider intercepts
Model 3	Add trigonometric terms for time of day
Model 4	Additive model with cubic cyclic spline for time of day
Model 5	Additive model with spline for ride length
Model 6	Remove random rider intercepts from Model 4

3.1 Six Models for Probability of a Negative Ride Rating

Model 1, which we will use as the baseline for comparing further models, is a multiple logistic regression model. Then, our first model will be,

$$\mathbb{P}(Y_i = 1) = \text{logit}^{-1}(\alpha + X_i\beta),$$

where $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}^p$ are parameters to be estimated. (X is the matrix of rider-level predictors specified at the end of Section 1.3.)

Riders appear to have different tendencies to rate rides negatively more often, as

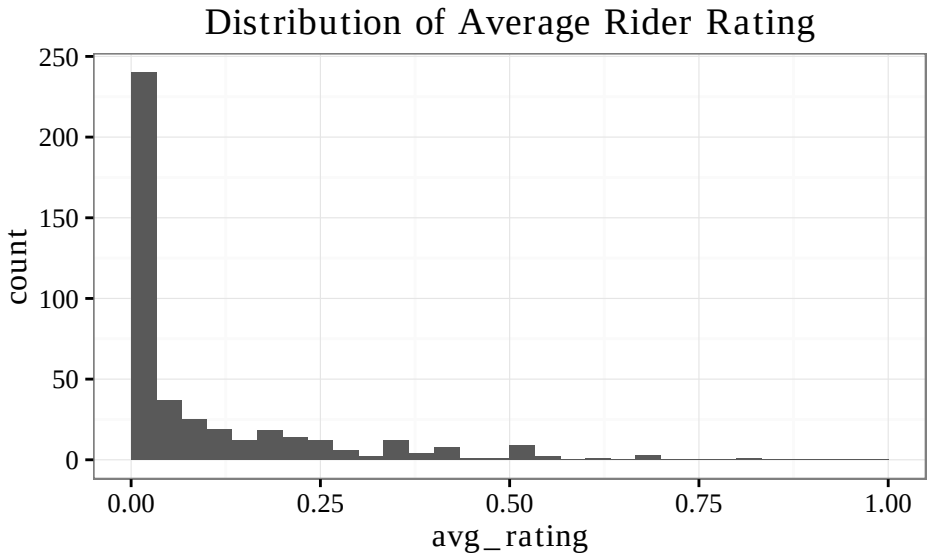

Figure 3.1: The overall rates at which each rider gives a negative rating for a ride varies greatly. This is our primary motivation for including rider intercepts and predictors.

we note in Figure 3.1. In fact, many riders give zero or nearly zero negative ratings. For **Model 2**, we account for this variability by adding intercepts that vary by rider:

$$y_i \sim \text{Bernoulli}\left(\text{logit}^{-1}\left(\alpha + \alpha_{j[i]} + X_i\beta\right)\right), \tag{3.1}$$

for $i = 1, \ldots, n$.

Rider intercepts themselves aren't as interesting as how they deviate from the mean, so we keep a fixed intercept α and constrain the rider intercepts, α_j, by specifying

$$\alpha_j \sim N(0, \sigma_\alpha^2).$$

Starting with **Model 3**, we address time of day, $t \in [0, 24)$ as a predictor. (We measure time of day in hours since midnight.) We use time of day to account for the all daily trends that may affect ratings, including as a simple way to model the overall traffic level, which is difficult to model on its own. These patterns are cyclic and very non-linear, so we can't time as a linear term. One approach is to add sinusoidal terms with a period of one day. We would be interested in fitting a term,

$$\beta \sin(Tx^{\text{time}} + \phi).$$

Estimating β wouldn't be hard: we can easily estimate coefficents of tranformed terms; it's more difficult to estimate T and ϕ. But, we know that we want to restrict our terms to fitting trends that happen over the course of one day, so we can set $T = 2\pi/d$, where d is 24 hours or some fraction of that.

As for ϕ, a trignometric transformation reframes the estimation of a phase shift parameter into the estimation of two coefficients for trignometric functions with no phase shift:

$$\begin{aligned}\beta \sin(Tx + \phi) &= \beta \left(\sin(Tx)\cos(\phi) + \cos(Tx)\sin(\phi)\right) \\ &= \beta \cos(\phi))\sin(Tx) + \sin(\phi)\cos(Tx) \\ &= \beta_1 \sin(Tx) + \beta_2 \cos(Tx),\end{aligned}$$

where $\beta_1 = \beta\cos(\phi)$ and $\beta_2 = \sin(\phi)$. At this point, we are now just estimating the coefficients of a couple of transformed variables, which can easily be done in any package that does generalized linear regressions.

We also want to take into account that weekday hourly patterns may be different than weekend patterns. We use a variable X^{weekend} that serves as a weekend indicator. For Model 3, we add two sets of sinusoidal terms: one set for weekdays and one for weekends. More explicitly, we define the model,

$$\begin{aligned}\mathbb{P}(Y_i = 1) = \text{logit}^{-1}(&\alpha + \alpha_{j[i]} + X_i\beta \\ &+ X^{\text{weekend}} \cdot [\beta^{t1}\sin(T \cdot t) + \beta^{t2}\cos(T \cdot t) \\ &+ \beta^{t3}\sin(T/2 \cdot t) + \beta^{t4}\cos(T/2 \cdot t)] \\ &+ (1 - X^{\text{weekend}}) \cdot [\beta^{t1}\sin(T \cdot t) + \beta^{t2}\cos(T \cdot t) \\ &+ \beta^{t3}\sin(T/2 \cdot t) + \beta^{t4}\cos(T/2 \cdot t))].\end{aligned} \tag{3.2}$$

For **Model 4** and **Model 5**, we abandon parametric methods and use a cyclic non-parametric smoother to model time of day. The only problem is that we need a way to combine our parametric and multilevel parts of the model with a new non-parametric part. This is where additive models come in.

Model 4 will introduce a cyclic cubic splines for time of day.

3.2 Model Evaluation

To fit the data, we got all of the rides in Portland, OR from December 3, 2014 to February 8, 2016 for riders that had over 20 rides. There were 35,370 rides, 14,032 of which were rated. Overall, 10.88 percent of these rides were given a negative rating. There were 518 riders in the data set.

The separation plots in **??** give a clear initial picture of how these model fits compare. Model 1 performs very poorly compared to those that include rider intercepts, assigning the same probability to most observations. Models that include the rider intercept perform similarly to each other. The log likelihoods and AIC scores, shown in **??**, corroborate this. Adding time dependency doesn't seem to impact predictive ability. We will see later, however, that it gives a fascinating result to interpret.

The gains from the rider intercepts are great, but we are compelled to ask: how much of that gain could have been achieved with randomly chosen groups? *i.e.* if riders were randomly assigned to rides, would the flexibility in the model created by allowing intercepts to vary increase predictive performance to the same degree? To test this, we ran a Model 4 after we randomly assign the rides a rider. This quick test nullified this skepticism, as you can see in the resulting separation plots in Figure 3.2.

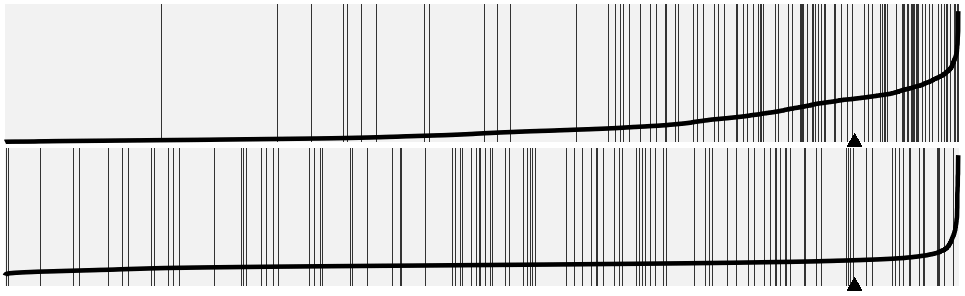

Figure 3.2: Separation plots for models 2 compared to a similar model where riders are randomly assigned to rides. This test demonstrates that the improvement in predictive performance provided by the rider intercepts was not coincidence.

3.3 Model Results

Table 3.2 presents the fixed effect estimates for our models.

The marginal fits for time of day, shown in **??**, are predictable. On weekdays, the probability of a negative rating peaks in the afternoon from 4–6 p.m., around when we expect rush hour traffic, and on weekends it stays steady throughout the day. While Model 4 and Model 5 give similar fits for time of day, Model 3's predictions peak at different times on weekdays and exhibit much more variability on weekends. There are two probable reasons for these differences: first, the sinusoidal terms are less flexibile than the splines; second, the splines, because they are non-parametric functions, penalize complexity of the fit while the parametric sinusoidal form does not, making the splines more conservative in their "curviness." The former explains the discrepancies in the weekday fits while the latter explains the discrepancies in the weekend fits. Given these differences, fitting time of day with splines is preferable; there is no motivation to constrain the functional form to any strict parametric form.

But these marginal time-of-day fits don't just tell a story about our time terms; they also reveal part of why the random rider intercepts are such powerful predictors. Notice that in comparing **??** to **??**, the scale at which the Model 6 time fitted probabilities vary is much larger than the scale at which the other models' predictions vary. (The differences is so large we had to show them in separate plots!) Without allowing for varying rider intercepts, the time terms take on a significant role. Yet, interestingly, the time term has nowhere near the amount of information that the rider intercepts seem to encode, according to the separation plots in **??**.

A clearer picture of what is going on is painted in Figure 3.4. The predictions of Model 1, which has a fixed intercept and no time dependence, don't vary by time of day. The models with random intercepts (2–5) show strong temporal patterns, with concentrated spikes in the morning and evening. Model 6, which had the time of day spline but a fixed intercept, has a temporal pattern, but does not represent the same degree time dependence that the random intercept models do.

One should be cautious about interpreting the intercepts. It's tempting to say they represent a riders general tendency to rate rides negatively, but this is ignoring their

Table 3.2: Regression coefficients for Model 1, Model 2, Model 4, and Model 6. 95% confidence intervals are given in parentheses.

Regression Term	Model 1	Model 2	Model 4	Model 6
Log(length)	-0.122	-0.100	-0.092	-0.114
	(-0.180, -0.063)	(-0.168, -0.032)	(-0.162, -0.022)	(-0.174, -0.054)
Mean Temp.	0.053	0.076	0.075	0.069
	(-0.0004, 0.110)	(0.005, 0.147)	(0.003, 0.147)	(0.012, 0.127)
Mean Wind speed	0.028	0.014	0.012	0.027
	(0.004, 0.052)	(-0.013, 0.041)	(-0.014, 0.039)	(0.002, 0.051)
Gust speed	-0.003	0.001	0.001	-0.003
	(-0.015, 0.008)	(-0.012, 0.013)	(-0.012, 0.013)	(-0.014, 0.009)
Rainfall	0.008	0.012	0.008	0.005
	(-0.015, 0.031)	(-0.015, 0.038)	(-0.019, 0.035)	(-0.019, 0.027)
Rainfall 4-Hour	0.013	0.016	0.017	0.014
	(0.005, 0.021)	(0.007, 0.025)	(0.008, 0.027)	(0.006, 0.022)
Intercept	-2.2868	-3.075	-3.127	-2.313
	(-2.428, -2.108)	(-3.386, -2.764)	(-3.436, -2.818)	(-2.475, -2.151)

typical route. Just like how figure Figure 3.4 demonstrates that riders have unique patterns of what times of days they take rides, riders also likely have very particular routes that make up the majority of their rides. Because our models do not take into account the route, the intercepts are likely encoding the information about a riders typical route.

The fact that riders have apparent patterns of time of ride (and very likely route of ride), indicates there will be some value in trying to differentiate the types of riders that are present in this data. Indeed, one way of looking at the range of probabilities of giving a negative rating different riders have (demonstrated in figure **??**) is that some are more likely to offer useful information. In the next chapter, we will attempt to pick apart different groups of riders, and attempt to characterize their patterns of when and how long they ride.

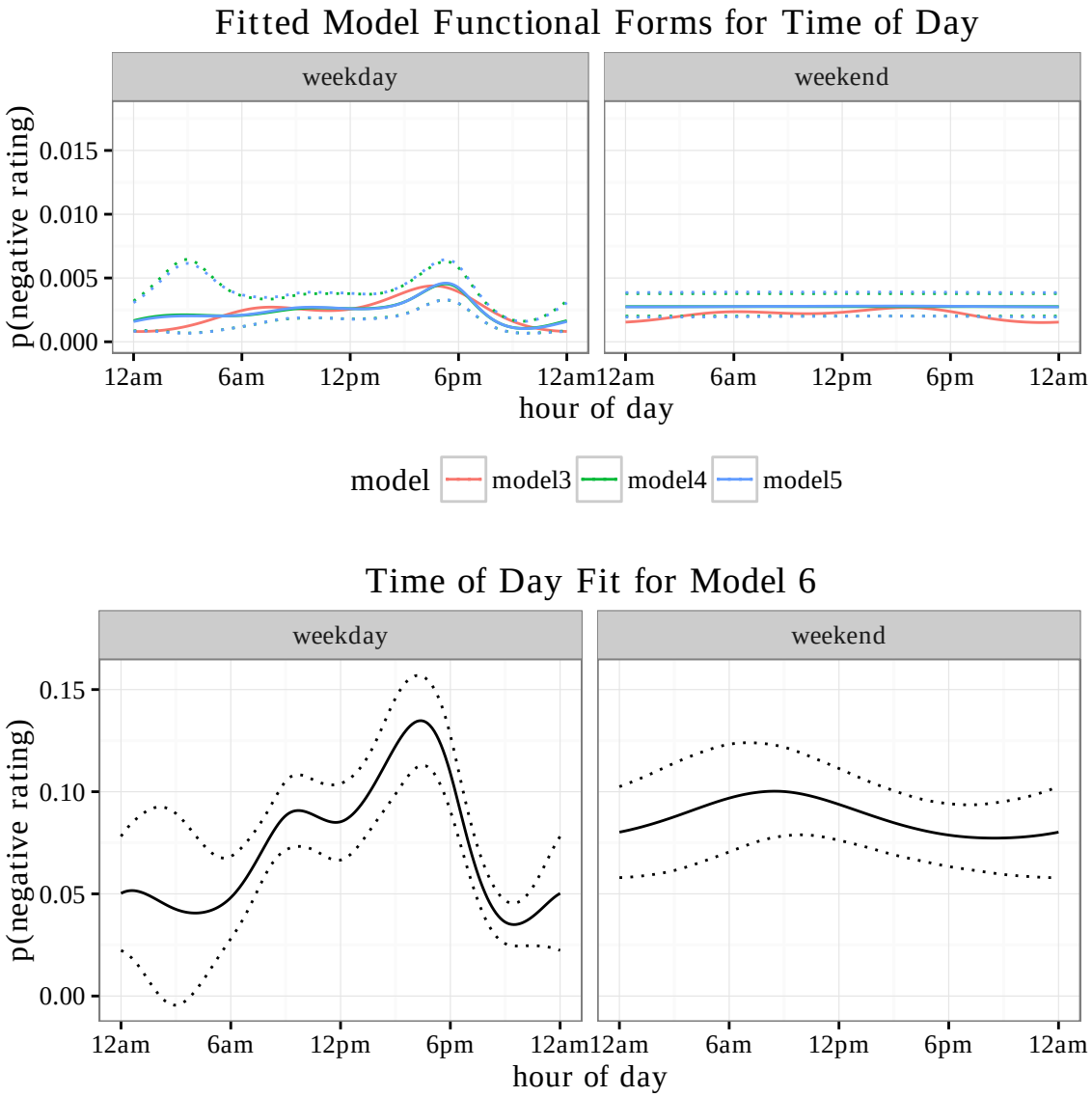

Figure 3.3: Predicted probabilities of a negative rating by time for a typical ride. The rider was chosen so the intercept was closest to the mean intercept for model 5. The median length and average mean temperature were used, and all other predictors were set to zero. The dotted lines show ±2 standard errors from the predictions.

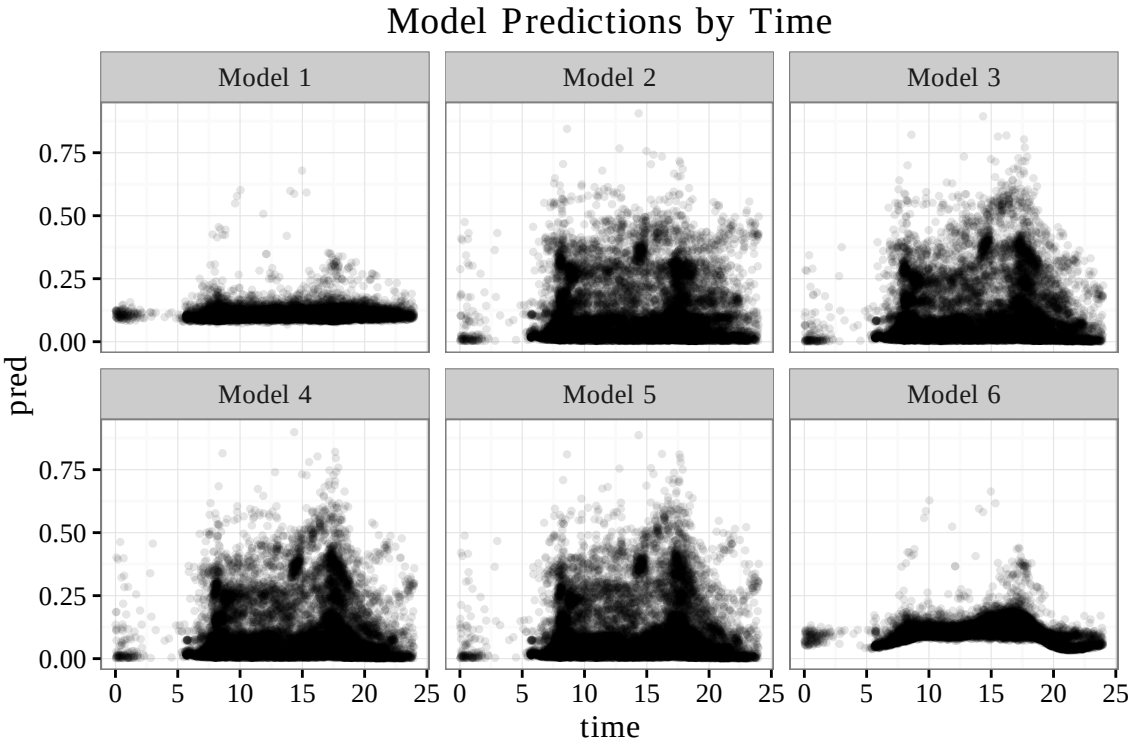

Figure 3.4: Each of the models predictions for all rides with time of day on the x-axis. Notice how starting with model 2, daily trends start to emerge. This indicates that the rider intercepts are picking up on time of day trends, which must be reflected in riders typical ride.

Chapter 4

Characterizing Riders

Given the results from last chapter, there is a clear need to understand what kinds of riders are in this data set. To do that, we identify predictors that differentiate types of riders and then use these variables to identify clusters of riders. We assess how these predictors do as rider-level predictors for the rider intercepts.

We also compare random intercepts with riders to random intercept models done by cluster. Ride Report, because they need to respect the privacy of their users, cannot identify individual riders in the data they provide to clients, yet our model results show that differentiating riders is crucial to getting good estimates in our models. If we can identify clusters of riders in the data set that give us nearly the same information as grouping by individuals did, these could be provided by Ride Report without nearly the same level of risks to user privacy as identifying individual riders.

4.1 Extracting Features and Determining Clusters

We characterized riders based on their rides because Ride Report does not collect data about their users besides their rides and email address. We limited our exploration to riders that had over 20 rated rides, in order to focus on riders who had been using the app for some time and had an identifiable pattern of rides.

Cyclists patterns in their rides are complex, particularly their time patterns. Computing their mean ride length for weekends was useful, but mean time of day for their rides does not capture anything meaningful. So we took care in selecting features that distinguished different rider patterns we saw when exploring the data.

First, we define the collection of cyclist j's rides as $H_j = \{i | j[i] = j\}$. This index set can be partitioned into the rides that occured on the weekend, $H_j^{\text{weekend}} = \{i | i \in H_j, x_i^{\text{weekend}} = 1\}$, and those that occurred on weekdays, $H_j^{\text{weekday}} = H_j \setminus H_j^{\text{weekend}}$. Then we define the following for each rider j: frequency of rides[1] (u_j^{freq}), proportion of rides on weekdays (u_j^{weekend}), median length of rides on weekdays ($u_j^{\text{med.len}}$) and

[1] We define frequency of a cyclist's rides as the number of rides divided by the difference between the time of the most recent ride and time of the first ride. (Units are arbitrary, because we standardized all of our rider-level variables.)

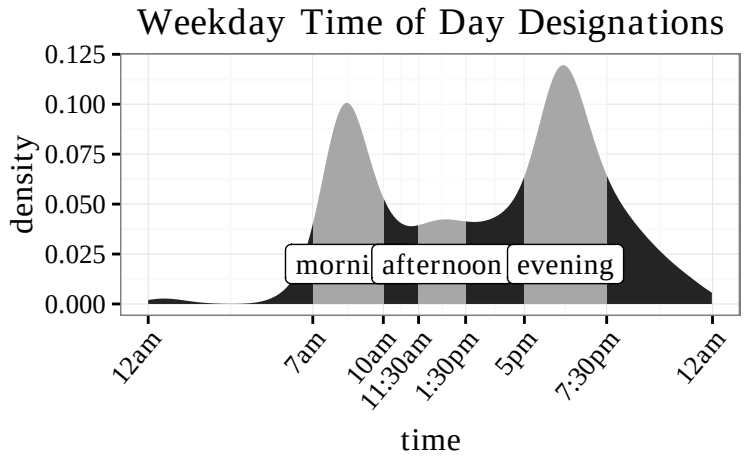

Figure 4.1: These intervals define the time designations we used in clustering. The proportion of each riders rides in each of these time intervals made up three of our features.

weekends ($u_j^{\text{med.len.w}}$), variance of ride length on weekdays ($u_j^{\text{var.len}}$) and weekends ($u_j^{\text{var.len.w}}$), and proportion of weekday rides during morning rush (u_j^{morning}), lunch rush (u_j^{lunch}), and evening rush (u_j^{evening}). The time intervals that describe the morning, lunch, and evening rush are shown in Figure 4.1.

Selecting variables for a cluster analysis is difficult, for the reason that many choices about what to use are arbitrary. We have chosen these variables, but two important other choices remain: what scales and transformations should these variables have? We chose here to transform all variables to be approximately gaussian—eliminating the right skew that was present in most of these features with log and square root transformations—and standardizing them by subtracting their mean and dividing by their standard deviation. Future research may find more appropriate ways to select features for clustering, but in our approach here we stick to a naive and simple approach to see what we can learn.

With the rider-level predictors in hand, we clustered the riders using k-means clustering. To choose the number of clusters, we assessed the total of the sum of squares within each cluster for different values of k, shown in Figure 4.2, and selected $k = 4$ as the point where we thought after which there was little value in having more clusters.

Looking at Figure 4.3, the clusters split the data into the four quadrants of the first two principal components of the rider data. This makes sense, given Figure 4.2 only showed small reductions in total within sum of squares as k got larger. Despite having failed to identify distinct clusters, studying these different groups still offers some valuable insight into how cylists differ in their ride patterns.

If we look at the length and time patterns of the clusters, shown in Figure 4.4, it's clear that these aggregate patterns give some distinct characterizations. Clusters 1 and 3 seem to be the groups that are the most consistent commuters, but are differentiated by the typical length of their weekend rides. Clusters 2 and 4 show

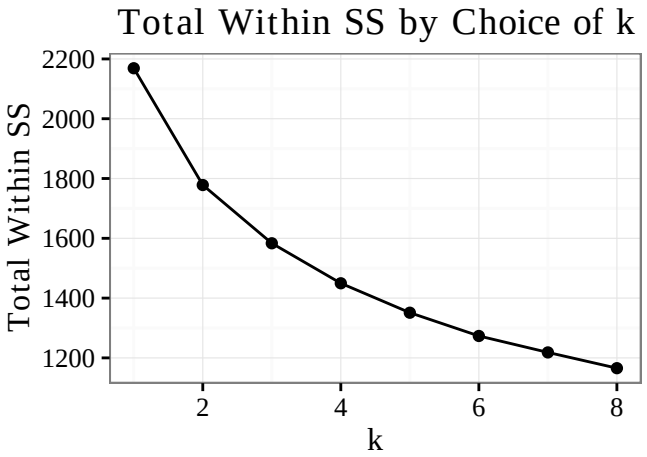

Figure 4.2: Total within sum of squares of each cluster, by number of clusters (k).

much more variance in the timing of their weekday rides, with cluster 2 having more consistently long weekend rides.

4.2 Models with Rider-Level Predictors

Having several rider-level predictors, we set out to see how well they predict rider intercepts. Now let U_j be the vector of rider-level variables. Then our model will be

$$Y_i \sim \text{Bernoulli}\left(\text{logit}^{-1}(\alpha_{j[i]} + X_i\beta)\right), \tag{4.1}$$

where,

$$\alpha_j \sim N(\gamma_0 + U_j\gamma, \sigma_\alpha), \tag{4.2}$$

with (γ_0, γ) being the group-level parameters we estimate.

This model should be comparable to Model 2, though because we are trying to get estimates of the rider-level predictor parameters, we make use of *Stan* instead of `lme4`.Though we would prefer to use a model similar to Model 4 from the previous chapter (the one with smoothing splines for time of day) the current additive mixed models package `gamm4` (which uses `lme4` to fit the mixed models part) does not support estimating the variability in group-level estimates. Unfortunately, in *Stan*, smoothing splines would have to be coded by hand and we lacked the expertise to write the functions to fit smoothing splines ourselves.

The ride-level predictor coefficients from the fitted model, are unimpressive. The variance in the rider intecepts not captured by the predictors, quantified with σ_α, is high.

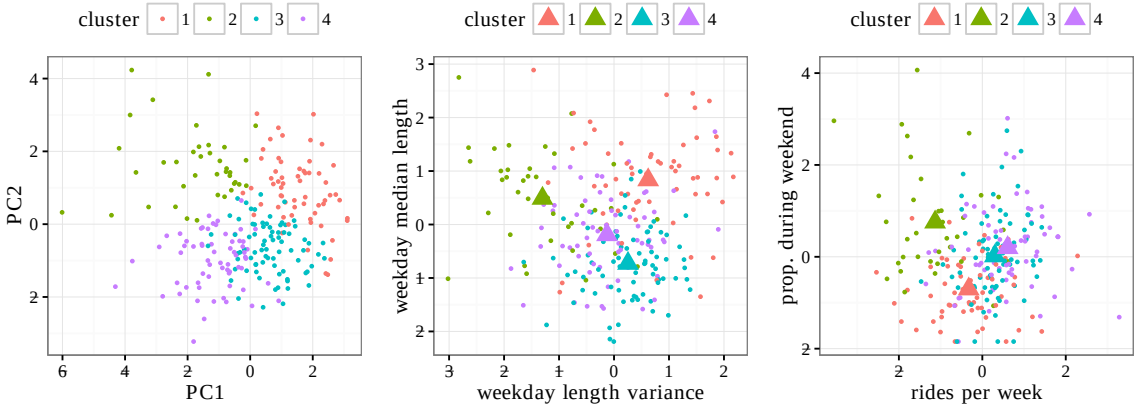

Figure 4.3: Rider clusters identified by k-means clustering. The triangles represent the centroids (computed as the mean) of the cluster members.

Table 4.1: Estimates of rider level predictors.

Parameter	Estimate	2.5% percentile	97.5% percentile
γ^{freq}	0.08	-0.19	0.35
γ^{weekend}	-0.13	-0.50	0.35
γ^{morning}	0.06	-0.22	0.34
$\gamma^{\text{afternoon}}$	0.13	-0.20	0.44
γ^{evening}	-0.02	-0.31	0.27
$\gamma^{\text{med.len}}$	0.01	-0.25	0.29
$\gamma^{\text{med.len.w}}$	0.08	-0.19	0.36
$\gamma^{\text{var.len}}$	0.07	-0.15	0.31
$\gamma^{\text{var.len.w}}$	-0.15	-0.47	0.17
γ_0	-2.99	-3.29	-2.69
σ_α	1.47	1.27	1.69

4.3 Cluster Intercepts Versus Rider Intercepts

Do these clusters provide similarly useful information that we got from introducing rider intercepts? Because a model with only 4 random intecepts—as in the case of cluster intercepts—rather than several hundred—as in the case with rider intercepts—is much less flexible, we expect that the cluster intercept model will preform much worse. One still might suspect there is still a significant benefit over a fixed intercept model.

There isn't. We computed Model 7, which is identical to Model 4 from the previous chapter, but has random intercepts by cluster rather than rider. Model 7 performed slightly better than Model 6—which only had a fixed intercept—but nowhere near as well as Model 4. The separation plots, $\log(\mathcal{L})$, AIC, and AUC measures, shown in **??**, all demonstrate this clearly. Given that the rider-level predictors did not seem to be

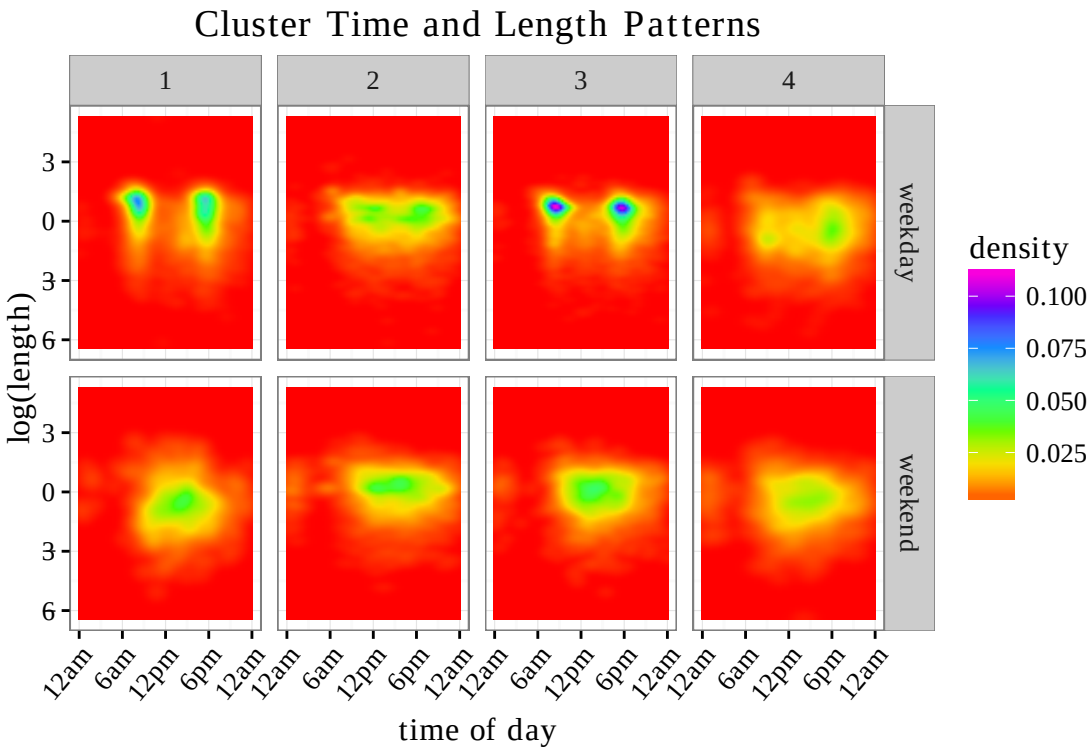

Figure 4.4: Patterns of ride length and ride time of day for each cluster..

predictive of the rider intercepts, this is not a surprise.

Chapter 5

Modeling Missing Response

We have a significant portion of observations with no response. But we do have all the predictors for every observation. So just as we built a model to predict the rating, can we build a model to predict whether a rider will give a rating? Going further, could we combine our rating model and nonresponse model into one model, such that our predictions for ratings will take into account biases in nonresponse?

We attempt to address these two questions in this chapter. The methods we present in this chapter could improve inference greatly on this data, but we do have concerns about the current quality of the data for which there are no reponses. As mentioned in Chapter 1, one big problem with data collection is that rides are often misclassified as bike rides when they are actually car rides or rides on public transit. We suspect that many of the unrated rides are rides that were misclassified as bike rides, and thus were not rated by the rider. (We assume that riders don't often go through the effort of correcting the classification of rides and know not to rate rides that weren't bike rides.) That being said, this issue could potentially be fixed and these methods used again in the future.

5.1 What could possibly go wrong?

We focus on the situation we have, where our response variable y_i has missing values. Define the vector $R = (r_1, r_2, \ldots, r_n)$ such that

$$r_i = \begin{cases} 1, & \text{if } y_i \text{ is missing;} \\ 0, & \text{if } y_i \text{ is observed;} \end{cases} \tag{5.1}$$

Rubin classifies missing data into three situations[1]:

1. **Missing Completely at Random (MCAR)**, where R is independent of Y and the predictors X. *i.e.*, $\mathbb{P}(R = 1|Y, X) = \mathbb{P}(R = 1.)$
2. **Missing at Random (MAR)**, where R is independent of Y, but may depend on X, *i.e.* $\mathbb{P}(R = 1|Y, X) = \mathbb{P}(R = 1|X)$
3. **Nonignorable, or not MCAR nor MAR**, where R is dependent on Y.

[1] Little and Rubin (1987) (page 14)

We believe that rider ratings may be correlated with nonresponse. One explanation: riders who have a unpleasant experience on the road might feel more incentive to report their experience on the app than those who had a ride that was uneventful. This motivates our exploration of missing data models, though, of course, is by no means evidence of nonignorable missing data.

If missing data is nonignorable, what could go wrong with our models? Let look at a toy example. Define the data set of n observations with $x \in \mathbb{R}^n$, $y \in \{0,1\}^n$, and R defined as before, where

$$x_i \sim \text{Normal}(0,1),$$

$$y_i \sim \text{Bernoulli}(\text{logit}^{-1}(4x_i)),$$

$$r_i \sim \text{Bernoulli}(0.3 + 0.4y_i),$$

for $i = 1, \ldots, n$.

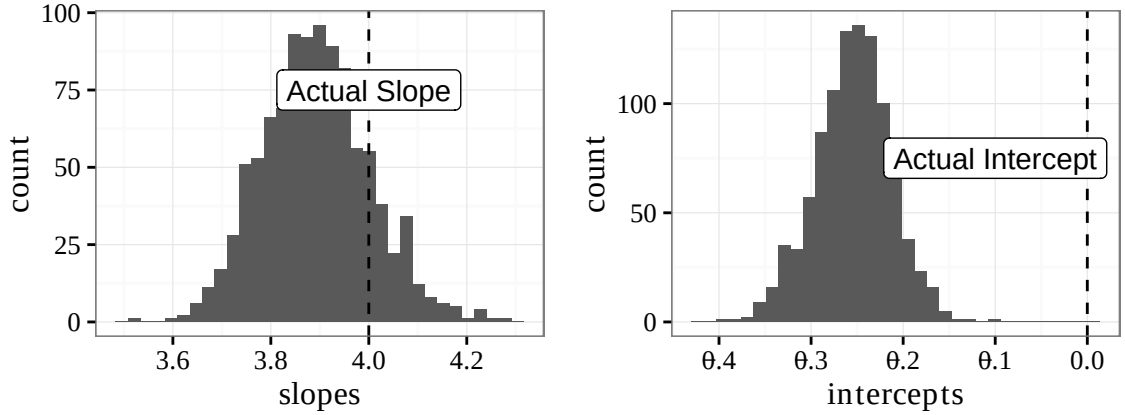

Figure 5.1: Simulated example of logistic regression fits to a model with nonignorable missing response. One data set of size $n = 10^4$ was computed from the toy data model. We then recomputed R 1,000 times, each time fitting a simple logistic regression model to Y and X.

If we attempt to fit a logistic regression model to this data, our slopes will be unbiased but our estimate of the intercept will be biased. Figure 5.1 shows the results of a simulation, computing the slope and intercepts for 1,000 different patterns of missing data for the same generated data set generated from our toy data model.

It makes sense that we are underestimating our intercept. The intercept can be interpreted as the base rate, and if values of $y_i = 1$ are more likely to be missing, the overall rate we observe will be lower.

Clearly, if we have nonignorable missing response, we are in a bad situation. Having missingness depend on Y leads to biased estimates of our intercepts when we fit models. But we do have all of our predictors of y, with no missingness. Could we leverage our understanding of how X predicts Y to understand the patterns of missing response?

5.2 Modeling the Missing Data Mechanism with Expectation Maximization

Here we perform the expectation maximization algorithm using the weighting method proposed by Ibrahim and Lipsitz[2]. Let y_i be our binary response and x_i be our predictors. With these we have our complete data logistic regression model $f(y_i \mid x_i, \beta)$, where β is a vector of parameters in the complete data model.

We then specify a logistic regression model for missingness (the r_i's): $f(r_i \mid x_i, y_i, \alpha)$, where α is the vector of parameters in the missingness model.

We begin the algorithm by getting our first estimates of α and β. We obtain $\beta^{(1)}$ by estimating β with only the non-missing data. We can then estimate the y_i for the missing data using $\beta^{(1)}$, and then use those estimates to compute $\alpha^{(1)}$.

For the E-step, we compute weights for each observation with missing response

$$w_{i\,y_i}^{(t)} = f(y_i \mid r_i, x_i, \alpha^{(t)}, \beta^{(t)}) = \frac{f(y_i \mid x_i, \beta^{(t)}) f(r_i \mid x_i, y_i, \alpha^{(t)})}{\sum_{y_i \in \{0,1\}} f(y_i \mid x_i, \beta^{(t)}) f(r_i \mid x_i, y_i, \alpha^{(t)})}. \tag{5.2}$$

For observed responses, $w_{i\,y_i}^{(t)} = 1$. Note that for each observation i, $\sum_{y_i \in \{0,1\}} w_{i\,y_i} = 1$. We can compute $f(y_i \mid x_i, \beta^{(t)})$ and $f(r_i \mid x_i, y_i, \alpha^{(t)})$ by making use of predictions from regression models. So in *R*, we can fit models and use the `predict()` function to get our probabilities from each of these models.

For the M-step, we find our next estimates of the parameters, $\alpha^{(t+1)}$ and $\beta^{(t+1)}$, by maximizing

$$Q(\alpha, \beta \mid \alpha^{(t)}, \beta^{(t)}) = \sum_{i=1}^{n} \sum_{y_i \in \{0,1\}} w_{iy_i}^{(t)} \cdot l(\alpha, \beta | x_i, y_i, r_i). \tag{5.3}$$

We do this by first by estimating $\beta^{(t+1)}$ using weighted maximum likelihood for the complete data model, and then estimating $\alpha^{(t+1)}$ using the same method. To maximize $l(\alpha, \beta | x_i, y_i, r_i)$, we maximize the product of their likelihoods,

$$l(\alpha, \beta | x_i, y_i, r_i) = l(\beta | x_i, y_i) l(\alpha | r_i, x_i, y_i),$$

which we can maximize by maximize each of the likelihoods separately because our estimates of α and β are only dependent on each other through x and y. This allows us to use any package that can fit models by maximum likelihood estimation using weights for the observations, which includes all of the model fitting packages we used in Chapter 3.

In order to create the data to fit these models, we create an augmented data set where each observation missing the response is recorded as two rows. These duplicate rows represent the two possible values of the response, and also contain the weights computed in the E-step. Figure 5.2 describes this process graphically.

We repeat the E and M step until the values of α and β converge.

[2]Ibrahim and Lipsitz (1996)

Figure 5.2: Creation of augmented data set for the weighted method of the EM algorithm for missing response data.

Original Data				**Augmented Data**			
y_i	x_i	r_i		y_i	x_i	r_i	w_i
1	2.4	0		1	2.4	0	1
0	1.3	0	$\rightarrow$	0	1.3	0	1
NA	-0.4	0		1	-0.4	0	0.2
				0	-0.4	0	0.8

As an example, we simulated a dataset from the same model we presented earlier of size 10^4. Of those observations, 6,252 were missing. As shown in Table 5.1, the estimate for the intercept in the model that only considers the complete data is way off, but the model resulting from the EM algorithm are nearly as accurate as the model computed fit to the full data (with missing values filled in from the original data model.) The missing data model is also able to get accurate estimates of the parameters that define the missing data mechanism, but the estimates are quite uncertain.

Table 5.1: Coefficients for models fit to simulated data set ($\pm$ twice the standard error.)

Model	$\hat{\beta}_0$	$2 \cdot SE_{\hat{\beta}_0}$	$\hat{\beta}_X$	$2 \cdot SE_{\hat{\beta}_X}$
Actual	0	–	4	–
Full Data Model	-0.009	0.065	3.881	0.080
Complete Data Model	-0.278	0.106	3.819	0.259
EM Final Model	0.042	0.065	3.814	0.157

Table 5.2: Estimates for missing data mechanism.

Model	$\hat{\alpha}_0$	$2 \cdot SE_{\hat{\alpha}_0}$	$\hat{\alpha}_Y$	$2 \cdot SE_{\hat{\alpha}_Y}$
Actual	0.3	–	0.4	–
EM Missing Data Model	0.263	0.132	0.530	0.268

5.3 Creating a Missing Data Model

In order to perform the algorithm, we need to specify a model for nonresponse. We will the same predictors that we do in Model 4 for ride rating—including a smoothing spline for time of day for weekdays and weekends—except we do not use random rider intercepts. We can specify a basic nonresponse model as

$$r_i \sim \text{Bernoulli}(\text{logit}^{-1}(\alpha + x_i\beta + \mathcal{S}(t_i))). \tag{5.4}$$

For the EM algorithm, we use Model 4 as our ride rating model and use the following model for the rating nonresponse mechanism:

$$r_i \sim \text{Bernoulli}(\text{logit}^{-1}(\alpha + y_i\beta_Y + x_i\beta + \mathcal{S}(t_i))). \tag{5.5}$$

5.4 EM Model Results

Table 5.3: Estimates for ride rating nonresponse mechanism.

Parameter	**Basic Nonresponse Model**	**EM Nonresponse Model**
y	0.730	1.035
	(0.235, 1.224)	(0.493, 1.577)
Log(Length)	-0.297	-0.327
	(-0.362, -0.232)	(-1.163, -0.771)
Mean Temperature	0.200	0.139
	(0.139, 0.262)	(0.077, -0.262)
Mean Wind Speed	0.032	0.031
	(0.003, 0.060)	(0.001, 0.061)
Max Gust Speed	-0.003	-0.007
	(-0.016, 0.010)	(-0.021, 0.006)
Rainfall	0.007	-0.024
	(-0.028, 0.041)	(-0.057, 0.009)
Rainfall 4-Hour	-0.002	0.010
	(-0.012, 0.009)	(-0.001, 0.021)
Intercept	-0.927	-0.967
	(-1.124, -0.729)	(-1.163, -0.771)

Table 5.4: Ride rating model estimates after EM algorithm

Parameter	**Model 4**	**EM Model**
Log(Length)	-0.147	0.205
	(-0.290, -0.005)	(-3.603, -2.684)
Mean Temperature	0.142	0.100
	(0.004, 0.281)	(0.005, 0.196)
Mean Wind Speed	0.002	-0.026
	(-0.054, 0.057)	(-0.069, 0.016)
Max Gust Speed	-0.005	0.020
	(-0.031, 0.021)	(0.001, 0.039)
Rainfall	0.050	0.051
	(-0.017, 0.117)	(0.009, 0.093)
Rainfall 4-Hour	0.022	0.017
	(0.003, 0.041)	(0.003, 0.030)
Intercept	-2.792	-3.144
	(-3.334, -2.250)	(-3.604, -2.684)

Chapter 6

Unfinished Work: Incorporating Routes

These models so far do not incorporate routes. Though our initial aim was to create models that incorporate route, we were not able to transform the route to a state that was useful for modeling. I leave the models as they are, but here I explain some of my work toward the goal of incorporating routes and describe some potential modeling approaches. Throughout this work is the caveat that much of these results are hard to interpret without taking into account route. We hope future researchers will be able to accomplish this.

6.1 Data Structures for Routes

Before one can model routes, they need to be represented properly. Ride Report's approach has been to discretize the routes into road segments by map matching the GPS traces to road segments in the Open Street Map (OSM) data. Road segments are convenient in that they are easily obtained and there are also accessible segment-level data, such as presence of bike lanes, types of roads, etc.

Road segments are not the only way to represent routes, however. One could also consider a route a sequence of intersections. Intersections may be the more stressful and dangerous portion of a riders route. We could also combine segments and intersections into a directed graph network.

6.2 Regression Terms for Road Segments

Now we have the task of incorporating our knowledge of riders' routes into our regression. Our approach here will be to consider routes as sequences of discrete road segments, each of which have known properties. This is convenient because we have such data about roads that give us bike lanes, road size, etc. It is even possible for us to calculate popularity of particular segments easily.

Assume we have K total road segments in our road network and for each ride we have $\Omega_i \subseteq \{1, \ldots, K\}$, the set of road segments that are in the route of ride i. Let l_k

be the length of the kth segment and define the length of ride i to be:

$$L_i = \sum_{k \in \Omega_i} l_k.$$

For the kth road segment, define the m-dimensional vector $W_k = W_k^1, W_k^2, \ldots, W_k^m$ road segment-level predictors. Then we shall define the term in our regression for the route of ride i as

$$R_i = \frac{1}{L_i} \sum_{k \in \Omega_i} l_k W_k \beta^{\text{road}},$$

Where β^{road} is a vector of coefficients for the road segment level predictors. When actually computing this value, it may be convenient to factor out the β^{road}

Conclusion

If we don't want Conclusion to have a chapter number next to it, we can add the `{.unnumbered}` attribute. This has an unintended consequence of the sections being labeled as 3.6 for example though instead of 4.1. The LaTeX commands immediately following the Conclusion declaration get things back on track.

More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendix A

A Code Sample of the EM Algorithm

Despite its attractive features, there are few explicit explination of how to actually program the EM algorithm for missing response using weights. The theoretical explination is contained in Section 5.2, but for the benefit of the reader, we lay out the practical implementation here.

For this example, we present the same code used for the simulation in Section 5.2. The data can be simulated with,

```
inv_logit <- function(x) 1 / (1 + exp(-x))
n <- 1e4
simulated_data <- data.frame(x = rnorm(n, 0, 1),
                             y = rbinom(n, 1, probs = inv_logit(0.3 + 0.4 * x)),
                             r = rbinom(n, 1, probs = inv_logit(-0.4 + 1.2 * y)))
simulated_data$y <- ifelse(simulated_data$r == 1, NA, simulated_data$y)
```

For convenience, we define the models once as functions, so we can use them more than once.

```
fit_model_r <- function(df) glm(r ~ x + y_pred, data = df, family = binomial)
fit_model_y <- function(df) glm(y ~ x, data = df, family = binomial)
```

First, we separate out the portions of the data that are complete and that are missing the response. To make our code clear and simple, we make use of the `dplyr` package.

```
data_complete <- simulated_data %>% filter(!is.na(y)) %>% mutate(weight = 1)
data_missing <- simulated_data %>% filter(is.na(y)) %>% mutate(weight = NA)
```

We then start the algorithm with our initial guesses at the model for `y` and `r`:

```
model_y <- fit_model_y(data_complete)
simulated_data$y_pred <- as.numeric(predict(model_y, newdata = simulated_data, type = "r
model_r <- fit_model_r(simulated_data)
```

```
for (i in 1:50) {
  print(model_y$aic)
  # get prob of 1
  pred_y <- predict(model_y, newdata = df_missing, type="response")

  # get prob missing given y
  pred_r_y1 <- predict(model_r,
                       newdata = mutate(df_missing, pred_y = 1),
                       type="response")
  pred_r_y0 <- predict(model_r,
                       newdata = mutate(df_missing, pred_y = 0),
                       type="response")

  # Make weights
  denom <- (pred_y * pred_r_y1) + ((1-pred_y) * pred_r_y0)
  w_y1 <- pred_y * pred_r_y1 / denom
  w_y2 <- (1-pred_y) * pred_r_y0 / denom

 # print(pred)
  df_augmented <- bind_rows(df_complete,
                        mutate(df_missing,
                               weight = w_y1,
                               y_obs = 1),
                        mutate(df_missing,
                               weight = w_y2,
                               y_obs = 0))
  model_y <- gam(y_obs ~ x, data = df_augmented, family = binomial,
                 weights = df_augmented$weight)
  model_r <- gam(psi ~ pred_y + x, data = df_augmented, family = binomial,
                   weights = df_augmented$weight)
}
```

References

Cressie, Noel, and Cristopher K. Wikle. 2011. *Statistics for Spatio-Temporal Data*. John Wiley & Sons.

Esarey, Justin, and Andrew Pierce. 2012. "Assessing Fit Quality and Testing for Misspecification in Binary-Dependent Variable Models." *Political Analysis* 20 (4): 480–500.

Gelman, Andrew, and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University Press, New York.

Greenhill, Brian, Michael D. Ward, and Audrey Sacks. 2011. "The Separation Plot: A New Visual Method for Evaluating the Fit of Binary Models." *American Journal of Political Science* 55 (4). Midwest Political Science Association: 991–1002.

Ibrahim, Joseph G., and Stuart R. Lipsitz. 1996. "Parameter Estimation from Incomplete Data in Binomial Regression When the Missing Mechanism Is Nonignorable." *Biometrics* 52 (3): 1071–8.

Little, Roderick J.A., and Donald B. Rubin. 1987. *Statistical Analysis with Missing Data*. John Wiley & Sons.

Mackaronis, Julia E., Donald S. Strassberg, Jeanne M. Cundiff, and Deanna J. Cann. 2013. "Beholder and Beheld: A Multilevel Model of Perceived Sexual Appeal." *Archives of Sexual Behavior*, October.

Shalizi, Cosma. 2013a. "Additive Models." `http://www.stat.cmu.edu/~cshalizi/uADA/13/lectures/ch09.pdf`.

———. 2013b. "Splines." `http://www.stat.cmu.edu/~cshalizi/uADA/13/lectures/ch08.pdf`.

Wood, Simon. 2006. *Generalized Additive Models: An Introduction with R*. CRC press.