Vanessa Chan, Stats M148 Homework 6

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(prophet)
```

```
## Loading required package: Rcpp
## Loading required package: rlang
##
```

```
## Attaching package: 'rlang'
##
## The following object is masked from 'package:data.table':
##
##     :=
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
##
## rstan version 2.32.6 (Stan version 2.32.2)
##
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
##
##
## Attaching package: 'rstan'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
data <- fread("dat_train1.csv")
head(data)
```

```
##    customer_id account_id ed_id      event_name     event_timestamp
## 1:    15849251  383997507     4 browse_products 2021-11-04 14:11:15
## 2:    15849251  383997507     4 browse_products 2021-11-04 14:11:29
## 3:    15849251  383997507     4 browse_products 2021-11-04 14:12:10
## 4:    15849251  383997507     4 browse_products 2021-11-04 14:12:21
## 5:    15849251  383997507     4 browse_products 2021-11-04 14:12:24
## 6:    15849251  383997507     2  campaign_click 2021-11-29 06:00:00
##    journey_steps_until_end
## 1:                       1
## 2:                       2
## 3:                       3
## 4:                       4
## 5:                       5
## 6:                       6
```

Cleaning from previous homeworks

```
data <- data.frame(data)
colnames(data) <- c("customer_id", "account_id", "ed_id", "event_name", "event_timestamp", "journey_step
```

```
# library(tidyverse)
dist_data <- distinct(data[+-6])

rand <- sample(unique(dist_data$customer_id)+ 10000)
random_sample <- dist_data %>%
  filter(customer_id %in% rand)

random_sample <- random_sample %>%
  group_by(customer_id,account_id) %>%
  mutate(steps = n())
```

Task 1

Time series first action

```
not_user <- c(15, 16, 17, 12, 13, 14, 37, 28)
end_data <- max(random_sample$event_timestamp)
rand1 <- random_sample %>%
  group_by(customer_id, account_id) %>%
  filter(max(event_timestamp) <= end_data -days(60)) %>%
  filter(!(ed_id %in% not_user))

daily <- rand1 %>%
  group_by(customer_id, account_id) %>%
  mutate(start = min(event_timestamp)) %>%
  distinct(customer_id, account_id, start) %>%
  group_by(start) %>%
  summarise(count = n_distinct(customer_id, account_id))

weekly <- rand1 %>%
  group_by(customer_id, account_id) %>%
  mutate(start = min(event_timestamp)) %>%
  mutate(year = year(start), week = week(start)) %>%
  distinct(customer_id, account_id, year, week) %>%
  group_by(year, week) %>%
  summarise(count = n_distinct(customer_id, account_id))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

```
monthly1 <- rand1 %>%
  group_by(customer_id, account_id) %>%
  mutate(start = min(event_timestamp)) %>%
  mutate(year = year(start), month = month(start)) %>%
```

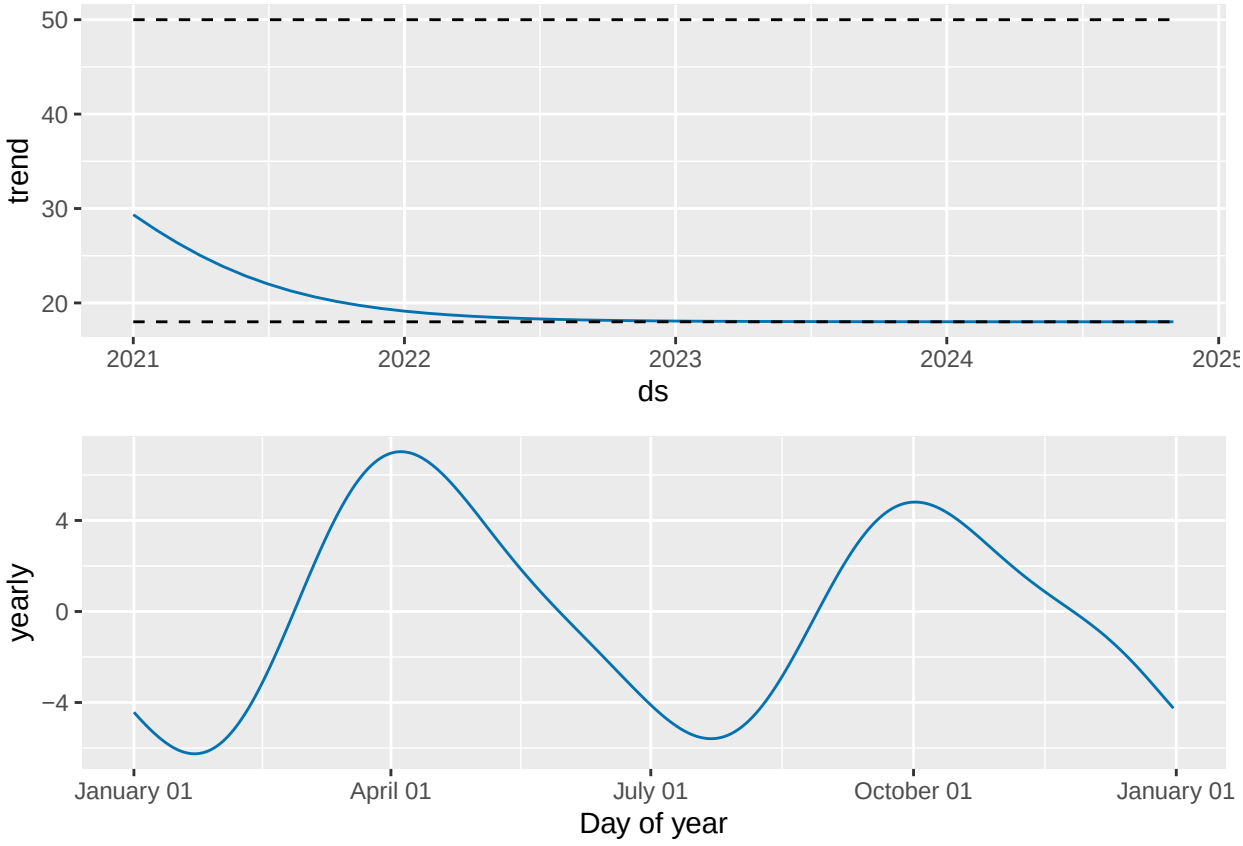
```
  distinct(customer_id, account_id, year, month) %>%
  group_by(year, month) %>%
  summarise(count = n_distinct(customer_id, account_id)) %>%
  filter( year != 2020)
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.
```

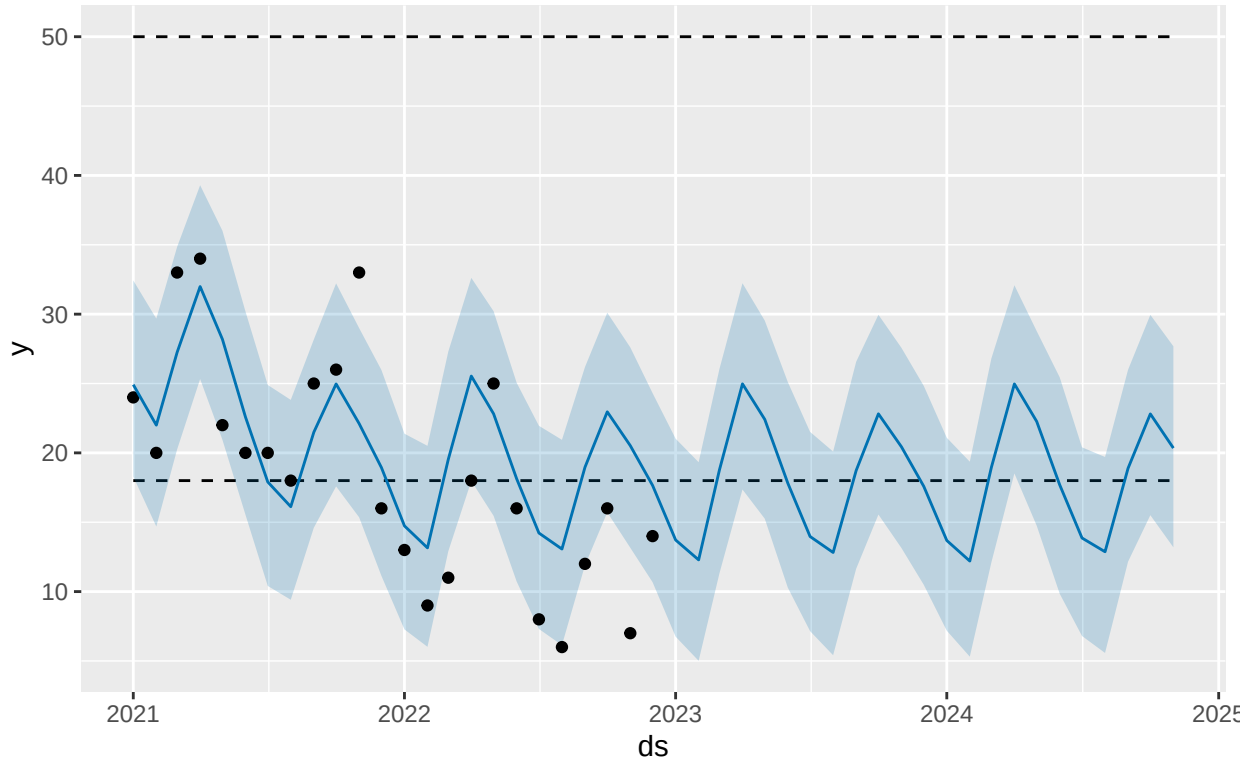
```
# impute missing december 2022 with december 2021 count
new_row <- data.frame(year = 2022, month = 12, count = 14)
monthly1 <- bind_rows(monthly1, new_row)
summary(monthly1$count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   12.75   18.00   18.58   24.25   34.00
```

```
month_dates <- character(0)
for (i in 1:nrow(monthly1)) {
  month_dates <- c(month_dates, paste(monthly1$year[i], monthly1$month[i], "01", sep = "-"))
}
month_dates <- as.Date(month_dates)
ts <- data.frame(ds = month_dates, cap = 50, floor = 18, y = monthly1$count)
ts_model <- prophet(ts, growth = "logistic", yearly.seasonality = 4, seasonality.mode = "additive",
                    n.changepoints = 0, weekly.seasonality = FALSE,
                    daily.seasonality = FALSE)

future_dates <- make_future_dataframe(ts_model, periods = 23, freq = "month")
future_dates <- data.frame(future_dates, floor = 18, cap = 50)
forecast <- predict(ts_model, future_dates)
prophet_plot_components(ts_model, forecast)
```




```
plot(ts_model, forecast)
```



The time series predicts that aside from the the seasonal pattern, there will be a slight decrease in started journeys from year to year. I am skeptical with this prediction, mostly because I believe the model is extrapolating a dip in users from 2021 to 2022, which could be due to the CCOVID-19 related economic downturn in 2022 leading users to not want to spend on more products, especially when Fingerhut's audience is already financialy vulnerable individuals. As the economy is recovering, I doubt this downward trend will continue.

But interestingly, we find that most users tend to start their user journeys around April and March, which is tax season. This is different from the pattern Jason found where most people have their orders shipped towards the holiday season. It is possible that while the holiday season stimulates existing users to buy presents on Fingerhut for their loved ones, most people discover Fingerhut during tax season when they are examining their own financial situation.

Task 2

Firstly, one thing I could have done to improve my model was to train the model on more data. Because my randomforest model (predicting on the first 5 steps) is so volatile, I was probably should have trained on a much larger set of data than the random sample we took. Additionally, I should have trained on the event "order placed" rather tha "order_shipped" as that is a user-motivated action and having "order_placed" in the data likely leads to overfitting as they are often directly correlated to each other. Lastly, I am still wondering if this entire path of predicting on the first 5 steps is even stable enough to be useful, and whether I should pivot to predicting on different variables instead.

Task 4: Survival Analysis

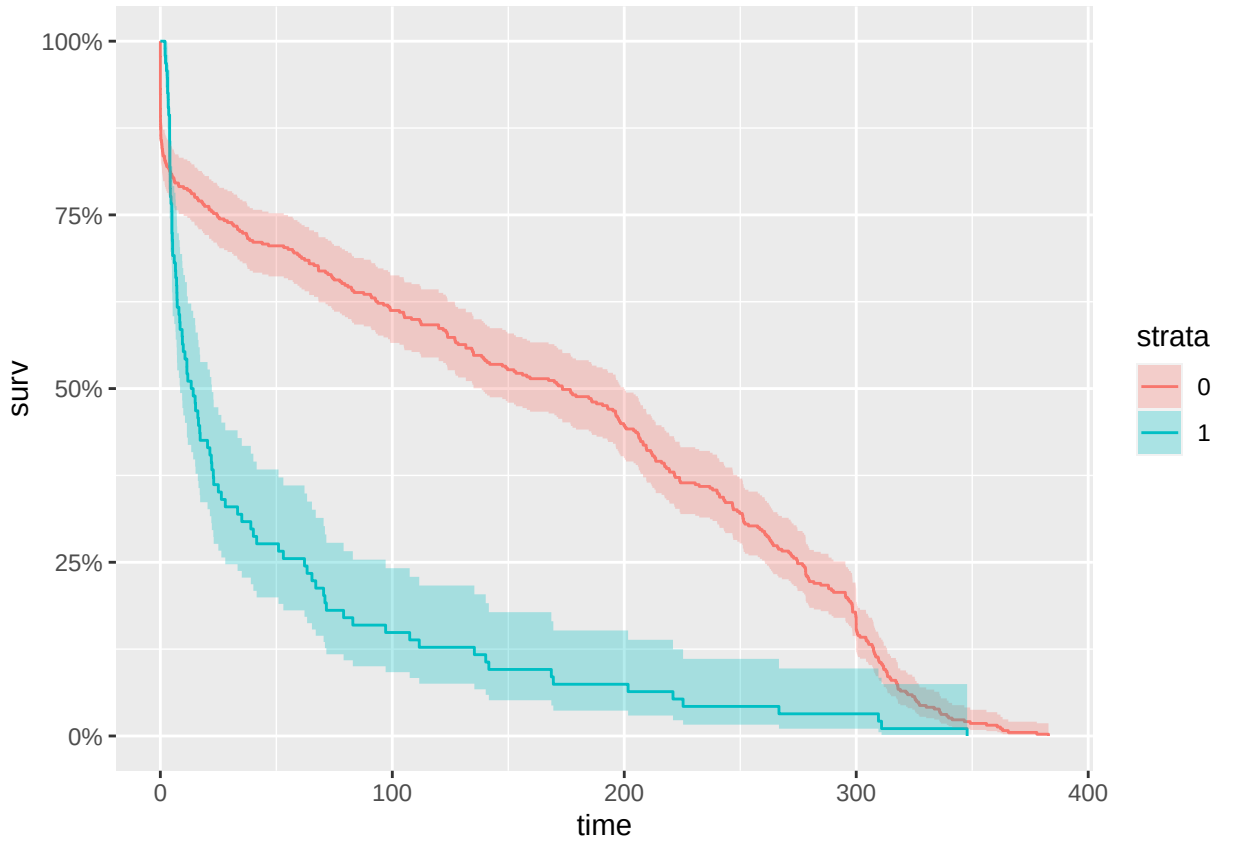
Here I constructed a survival analysis of when someone shipped an order, so a journey "dying out" means it is has been successfully converted into a order_shipped.

```
ed_ids_use <- c(1,3,4,5,6,11,19,21,24,28,29)

survivor <- random_sample %>%
  filter(ed_id %in% ed_ids_use) %>%
  group_by(account_id, customer_id) %>%
  mutate(length = max(event_timestamp) - min(event_timestamp)) %>%
  select(-event_timestamp) %>%
  group_by(account_id, customer_id, event_name) %>%
  mutate(num= n()) %>%
  ungroup() %>%
  distinct() %>%
  select(customer_id,account_id, event_name, length, num, steps) %>%
  pivot_wider(
    names_from = event_name,
    values_from = num
  ) %>%
  mutate_all(~ replace_na(., 0))
survivor <- survivor %>%
  mutate(length = as.numeric(length / 60 / 60 /24 )) %>%
  mutate(all_die = 1)

#install.packages("survival")
library(survival)
#install.packages("ggfortify")
library(ggfortify)

km <- with(survivor, Surv(length, order_shipped))

km_fit <- survfit(Surv(length, all_die) ~ order_shipped, data=survivor)
summary(km_fit, times = c(1,30,60,90*(1:10)))
```

```
## Call: survfit(formula = Surv(length, all_die) ~ order_shipped, data = survivor)
##
##                 order_shipped=0
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    326      61   0.8424 0.01852      0.80684       0.8795
##    30    286      40   0.7390 0.02232      0.69653       0.7841
##    60    267      19   0.6899 0.02351      0.64535       0.7376
##    90    246      21   0.6357 0.02446      0.58948       0.6855
##   180    189      57   0.4884 0.02541      0.44103       0.5408
##   270    103      86   0.2661 0.02247      0.22557       0.3140
##   360      6      97   0.0155 0.00628      0.00701       0.0343
##
##                 order_shipped=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1     94       0   1.0000  0.0000       1.0000       1.0000
##    30     31      63   0.3298  0.0485       0.2472       0.4399
##    60     24       7   0.2553  0.0450       0.1808       0.3606
```

```
##     90      15       9   0.1596  0.0378        0.1003        0.2538
##    180       7       8   0.0745  0.0271        0.0365        0.1519
##    270       3       4   0.0319  0.0181        0.0105        0.0972
```

```
autoplot(km_fit)
```




```
survivor <- survivor %>%
  mutate(browse_products_b = as.numeric(browse_products !=0)) %>%
  mutate(view_cart_b = as.numeric(view_cart!=0)) %>%
  mutate(begin_checkout_b = as.numeric(view_cart!=0)) %>%
  mutate(campaignemail_clicked_b = as.numeric(campaignemail_clicked!=0) )
```

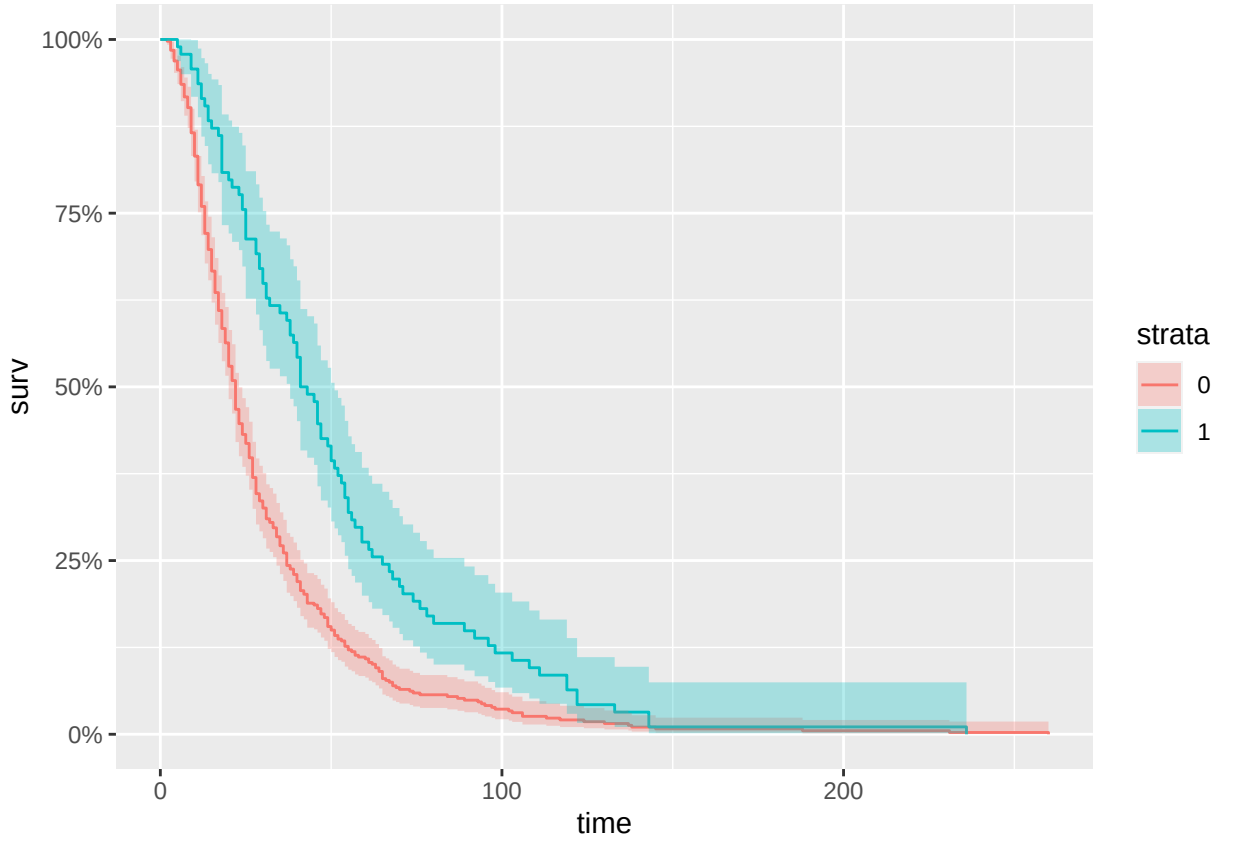
In this survival analysis, the red line symbolizes all active journeys and the blue line symbolizes successful journeys. The x-axis is based on number of days they were active on the site. Here, we can see that out of all the journeys that did manage to become successes, most of them (75%) died out at just over 50 days. The huge visual split then indicates that the number of days is a significant factor in predicting someone's success.

```
km <- with(survivor, Surv(steps, order_shipped))

km_fit <- survfit(Surv(steps, all_die) ~ order_shipped, data=survivor)
summary(km_fit, times = c(1,30,60,90*(1:10)))
```

```
## Call: survfit(formula = Surv(steps, all_die) ~ order_shipped, data = survivor)
```

```
##
##                 order_shipped=0
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    387       0  1.00000 0.00000       1.00000        1.0000
##    30    130     261  0.32558 0.02382       0.28209        0.3758
##    60     43      84  0.10853 0.01581       0.08157        0.1444
##    90     19      23  0.04910 0.01098       0.03167        0.0761
##   180      3      16  0.00775 0.00446       0.00251        0.0239
##
##                 order_shipped=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1     94       0   1.0000  0.0000       1.00000        1.0000
##    30     63      33   0.6489  0.0492       0.55928        0.7530
##    60     26      35   0.2766  0.0461       0.19946        0.3836
##    90     14      12   0.1489  0.0367       0.09186        0.2415
##   180      1      13   0.0106  0.0106       0.00151        0.0747
```

```
autoplot(km_fit)
```



This survival analysis is conflicting, showing that there is a higher proportion of orders_shipped at each number of steps than there are total journeys. I'm not sure why this graph isn't able to work given that the other survival analysis based on journey length seems to be interpreted normally. Hence, from here on out, variable analysis will be done using length.

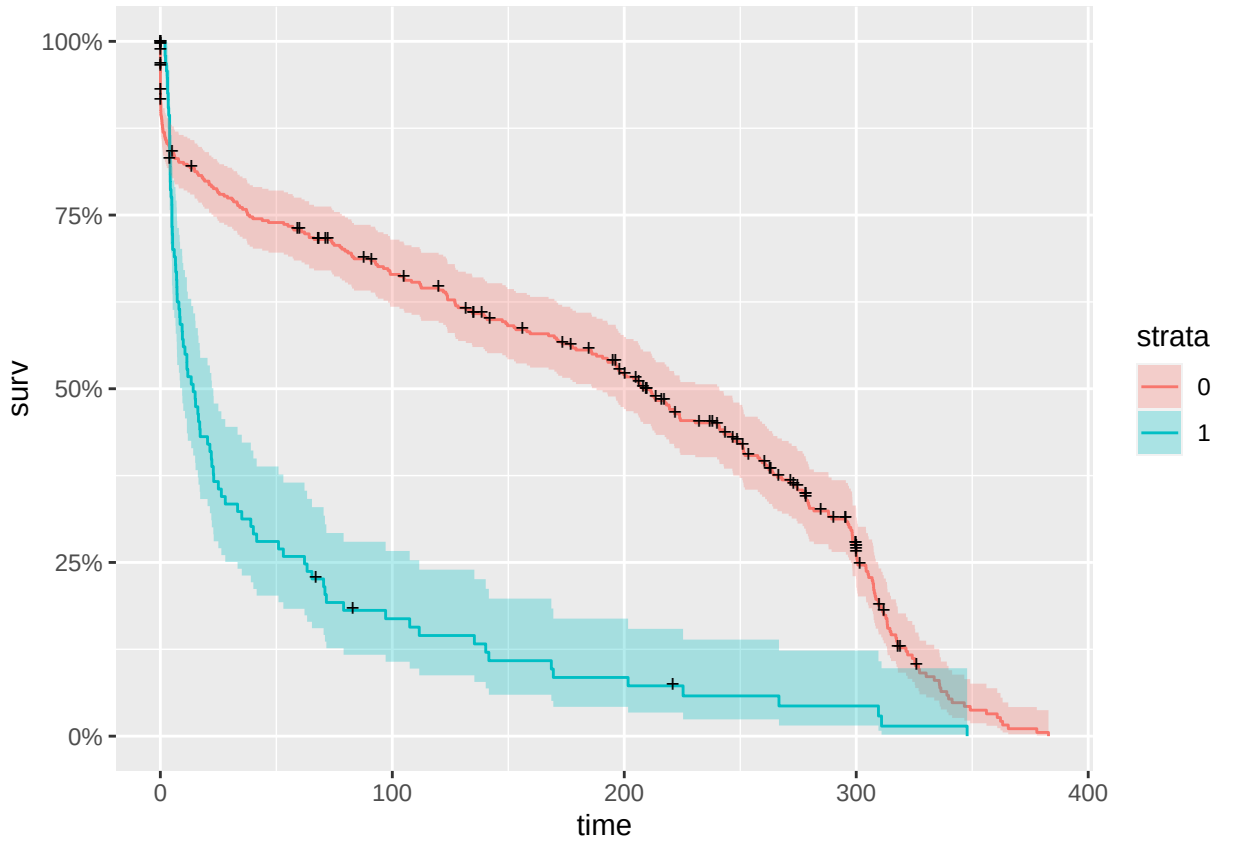
Next, I conducted some variable analysis using a boolean of each action (0 or 1) to symbolize whether the user had done that action or not. It would have been nice to also insert some numerical correlation (whether a user does that action more) but I'm not sure that can be done with the survivor package. ### Browse

Products

```
km <- with(survivor, Surv(length, order_shipped))

km_fit <- survfit(Surv(length, browse_products_b) ~ order_shipped, data=survivor)
summary(km_fit, times = c(1,30,60,90*(1:10)))
```

```
## Call: survfit(formula = Surv(length, browse_products_b) ~ order_shipped,
##     data = survivor)
##
##                 order_shipped=0
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    326      46   0.8770  0.0170       0.8444        0.911
##    30    286      38   0.7744  0.0217       0.7331        0.818
##    60    267      17   0.7284  0.0231       0.6846        0.775
##    90    246      15   0.6870  0.0241       0.6413        0.736
##   180    189      46   0.5558  0.0262       0.5068        0.609
##   270    103      58   0.3688  0.0266       0.3202        0.425
##   360      6      75   0.0321  0.0125       0.0149        0.069
##
##                 order_shipped=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1     94       0   1.0000  0.0000       1.0000        1.000
##    30     31      62   0.3341  0.0489       0.2507        0.445
##    60     24       7   0.2586  0.0454       0.1833        0.365
##    90     15       7   0.1810  0.0402       0.1172        0.280
##   180      7       8   0.0845  0.0299       0.0422        0.169
##   270      3       3   0.0435  0.0231       0.0153        0.123
```

```
autoplot(km_fit)
```



Here we can see that out of the people who did browse products (blue) tended to have their orders shipped earlier than those who did not (red), seeing that their journey "died off" faster. This means that browsing products is a good predictor of whether someone will place and order. As you can see, those who did browse products are 50% more likely to place an order than those who did not.

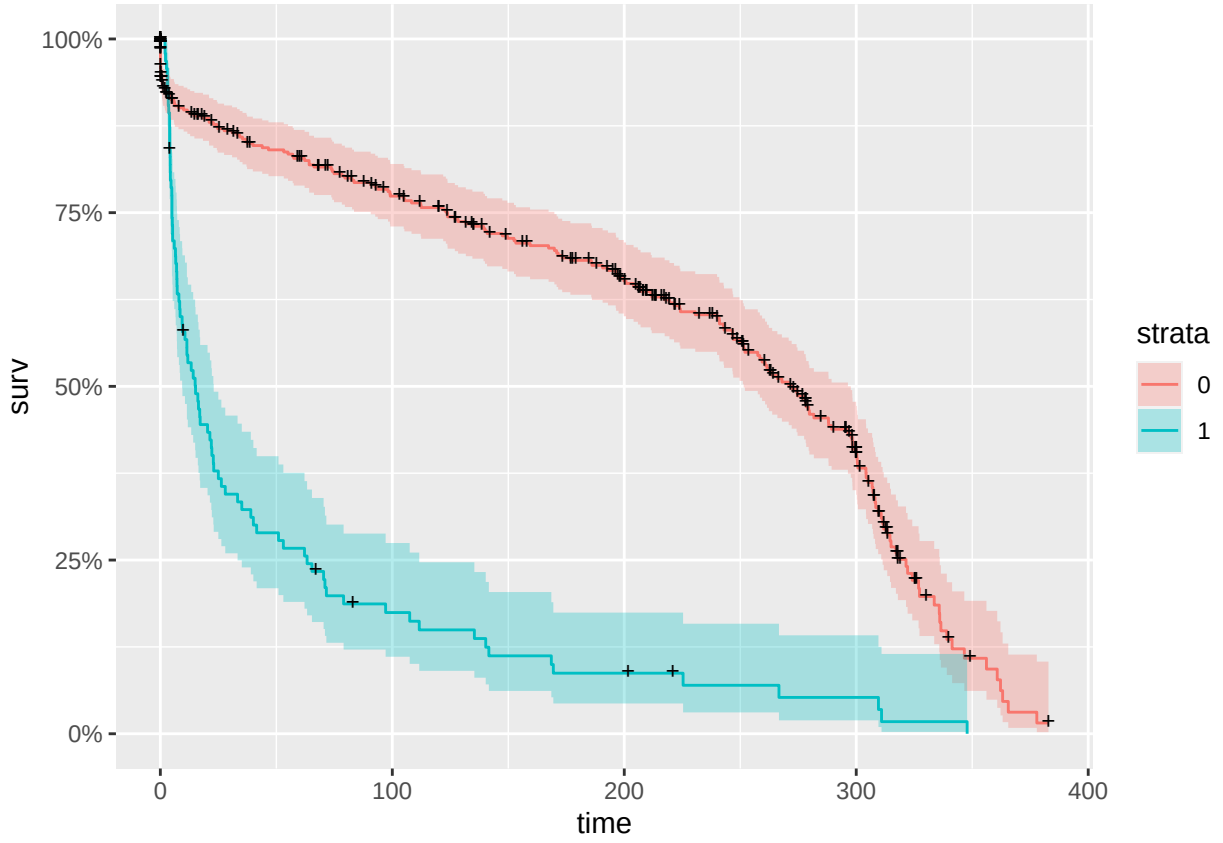
View Cart

```
km <- with(survivor, Surv(length, order_shipped))

km_fit <- survfit(Surv(length, view_cart_b) ~ order_shipped, data=survivor)
summary(km_fit, times = c(1,30,60,90*(1:10)))
```

```
## Call: survfit(formula = Surv(length, view_cart_b) ~ order_shipped,
##     data = survivor)
##
##                 order_shipped=0
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    326      23   0.9356  0.0130        0.911        0.961
##    30    286      23   0.8679  0.0182        0.833        0.904
##    60    267      13   0.8281  0.0204        0.789        0.869
##    90    246      11   0.7933  0.0221        0.751        0.838
##   180    189      33   0.6813  0.0262        0.632        0.735
##   270    103      41   0.5061  0.0308        0.449        0.570
##   360      6      52   0.0931  0.0305        0.049        0.177
##
```

```
##                 order_shipped=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1     94       0   1.0000  0.0000       1.0000        1.000
##    30     31      60   0.3449  0.0499       0.2597        0.458
##    60     24       7   0.2670  0.0465       0.1898        0.376
##    90     15       7   0.1869  0.0413       0.1212        0.288
##   180      7       8   0.0872  0.0308       0.0436        0.174
##   270      3       2   0.0523  0.0266       0.0193        0.142
```

```
autoplot(km_fit)
```



Here we can see that out of the people who did view_cart (blue) tended to have their orders shipped earlier than those who did not (red), seeing that their journey "died off" faster. This is expected behavior, given that those who view their cart are likely checking their order before they place it. My purpose with this graph it to check whether a significant amount of people might be viewing their carts and then deciding not to purchase after seeing the final amount. This seems to not be the case, which is healthy for Fingerhut.

Begin Checkout

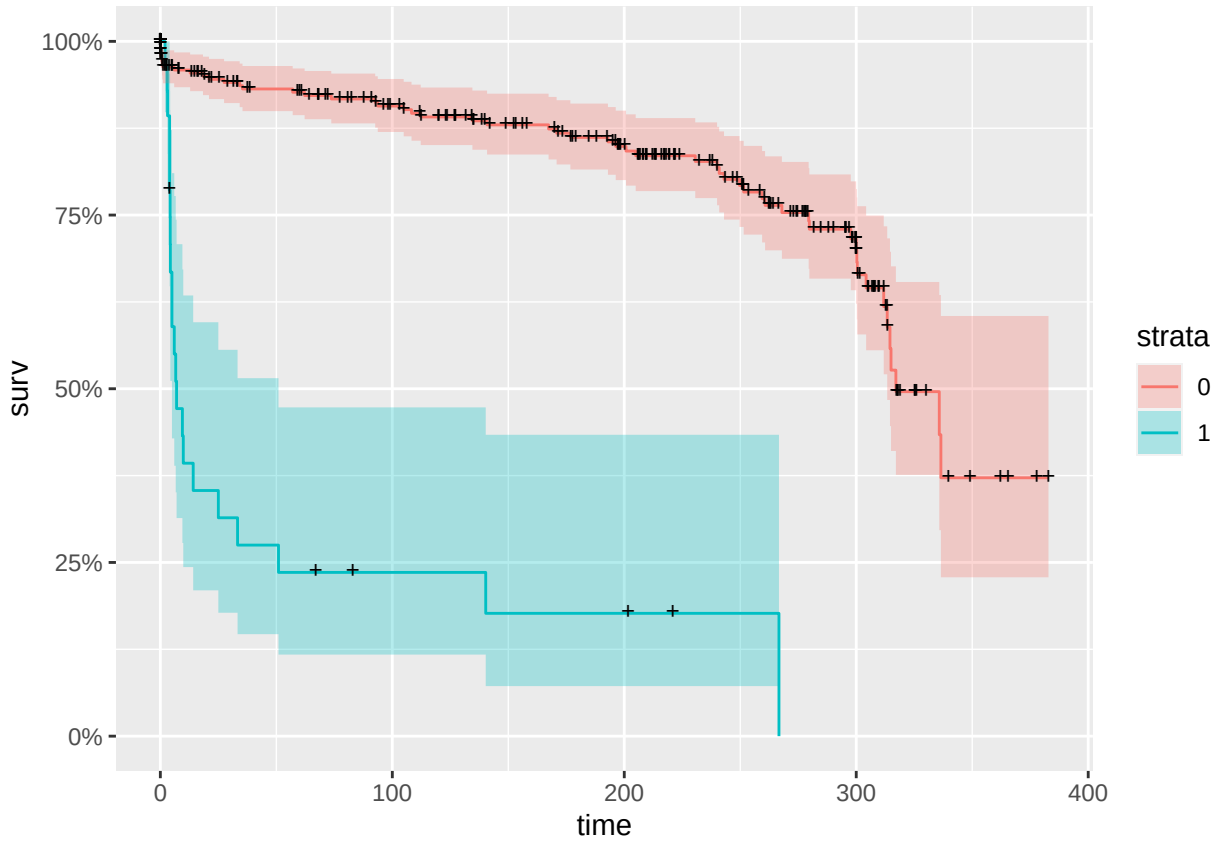
```
km <- with(survivor, Surv(length, order_shipped))

km_fit <- survfit(Surv(length, begin_checkout) ~ order_shipped, data=survivor)
```

```
## Warning in Surv(length, begin_checkout): Invalid status value, converted to NA
```

```
summary(km_fit, times = c(1,30,60,90*(1:10)))
```

```
## Call: survfit(formula = Surv(length, begin_checkout) ~ order_shipped,
##     data = survivor)
##
## 171 observations deleted due to missingness
##                 order_shipped=0
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    230       8    0.967  0.0114        0.945        0.990
##    30    205       6    0.941  0.0154        0.911        0.971
##    60    195       3    0.927  0.0171        0.894        0.961
##    90    182       2    0.917  0.0183        0.882        0.954
##   180    138      10    0.862  0.0242        0.816        0.910
##   270     73      13    0.754  0.0355        0.687        0.826
##   360      4      14    0.372  0.0923        0.229        0.605
##
##                 order_shipped=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1     28       0    1.000  0.0000        1.000        1.000
##    30      8      18    0.314  0.0915        0.178        0.556
##    60      6       2    0.236  0.0838        0.117        0.473
##    90      4       0    0.236  0.0838        0.117        0.473
##   180      3       1    0.177  0.0810        0.072        0.434
```

```
autoplot(km_fit)
```



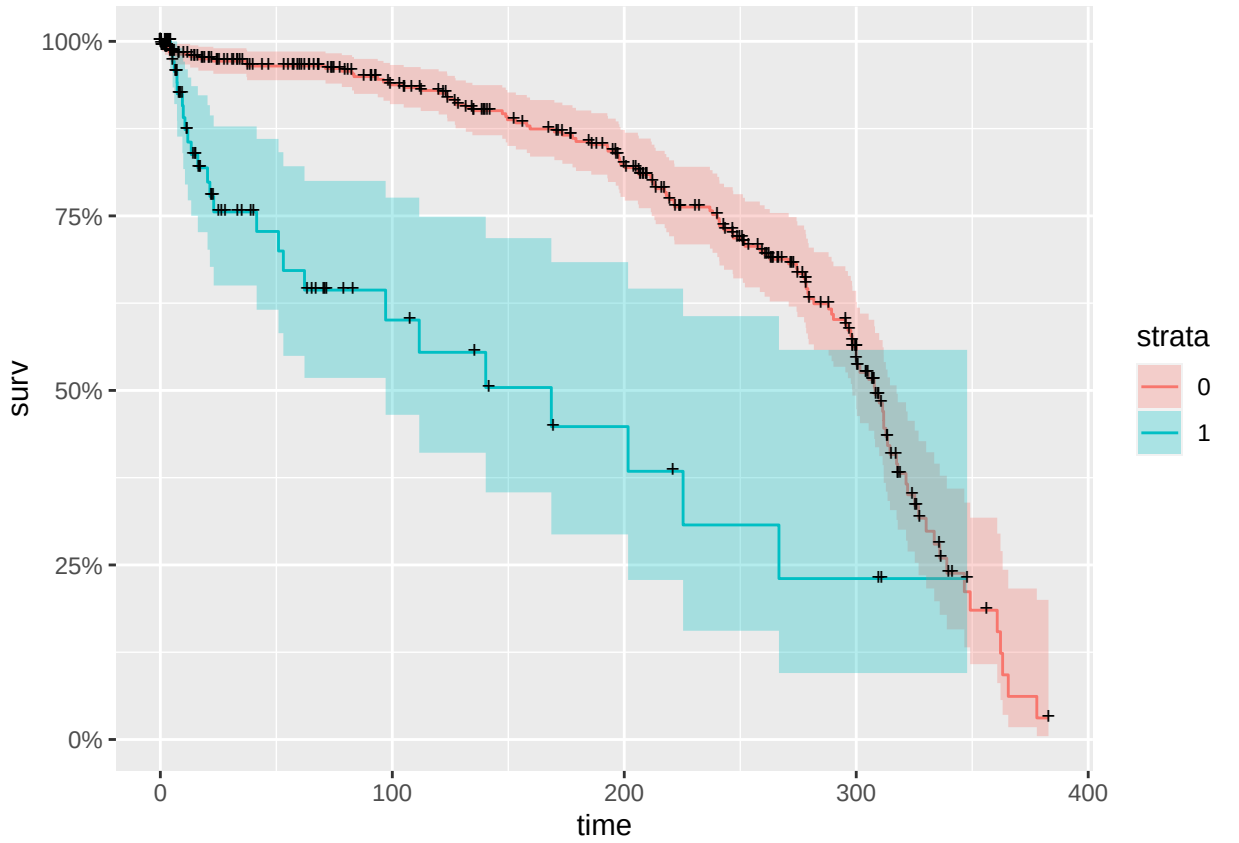
Here we can see that out of the people who did begin_checkout (blue) tended to have their orders shipped earlier than those who did not (red), seeing that their journey "died off" faster. This is expected behavior, given that those begin checkout are likely to complete the process. My purpose with this graph it to check whether a significant amount of people might be beginning checkout and then deciding not to purchase after seeing the final amount. This seems to be not the case, and especially because it seems like 100% of people who began checkout did eventually finish the checkout, even if it is 300 days after they began it.

Click Campaign Email

```
km <- with(survivor, Surv(length, order_shipped))

km_fit <- survfit(Surv(length, campaignemail_clicked_b) ~ order_shipped, data=survivor)
summary(km_fit, times = c(1,30,60,90*(1:10)))
```

```
## Call: survfit(formula = Surv(length, campaignemail_clicked_b) ~ order_shipped,
##     data = survivor)
##
##                 order_shipped=0
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1    326       2    0.994 0.00424        0.986        1.000
##    30    286       7    0.972 0.00934        0.953        0.990
##    60    267       2    0.965 0.01048        0.944        0.985
##    90    246       4    0.949 0.01278        0.925        0.975
##   180    189      22    0.857 0.02211        0.814        0.901
##   270    103      32    0.688 0.03230        0.627        0.754
##   360      6      43    0.185 0.05101        0.108        0.318
##
##                 order_shipped=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##     1     94       0    1.000  0.0000       1.0000        1.000
##    30     31      14    0.756  0.0579       0.6503        0.878
##    60     24       3    0.672  0.0688       0.5495        0.821
##    90     15       1    0.644  0.0714       0.5179        0.800
##   180      7       4    0.448  0.0966       0.2937        0.684
##   270      3       3    0.230  0.1040       0.0952        0.558
```

```
autoplot(km_fit)
```



Here we can see that out of the people who did click on the campaign email (blue) tended to have their orders shipped earlier than those who did not (red), seeing that their journey "died off" faster when the order was shipped. This graph shows that people who did click the campaign email were about 20-30% more likely to have their orders shipped than those who did not click the campaign email at the same point in time.