hw3

Will Tirone

Q1)

Load Data

First we load the data and extract things like the numeric representation of the day of the week and the month that we will use as covariates later.

```
brook = read.csv('data/Brooklyn-count.csv') |>
  mutate(Date = dmy(Date),
         num_day = as.factor(wday(Date)),
         day = wday(Date, label=TRUE),
         dom = day(Date),
         num_month = as.factor(month(Date)),
         month = month(Date, label=TRUE))

eye = read.csv('data/eyewitness.csv') |>
  mutate(Y0 = as.numeric(id == 'correct'),
         Y1 = as.numeric(id == 'foil'),
         Y2 = as.numeric(id == 'reject'))
```

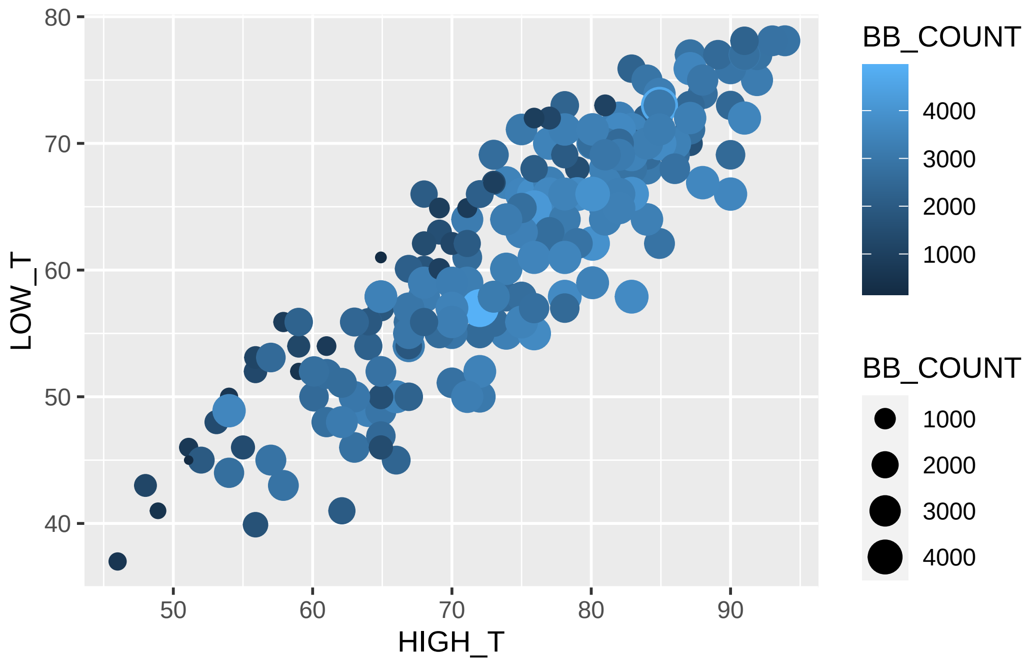
EDA

First, we'll take a look at the temperatures and size them by the count of cyclists. At low LOW_T and HIGH_T, we see fewer cyclists as expected. We can also see that at a given LOW_T, a higher HIGH_T results in more cyclists.

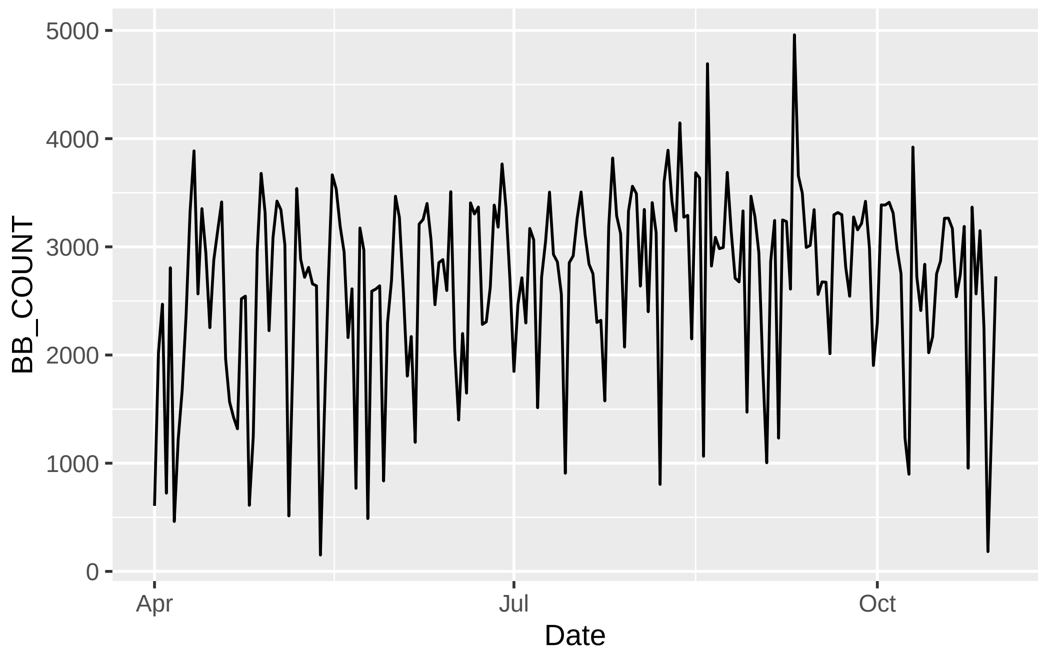
There is not an obvious time trend from April to October since the mean looks relatively constant. However, there is strong seasonality among the days and weeks of the months.

And just visually it looks like April has the widest ranges of biker count across different days.

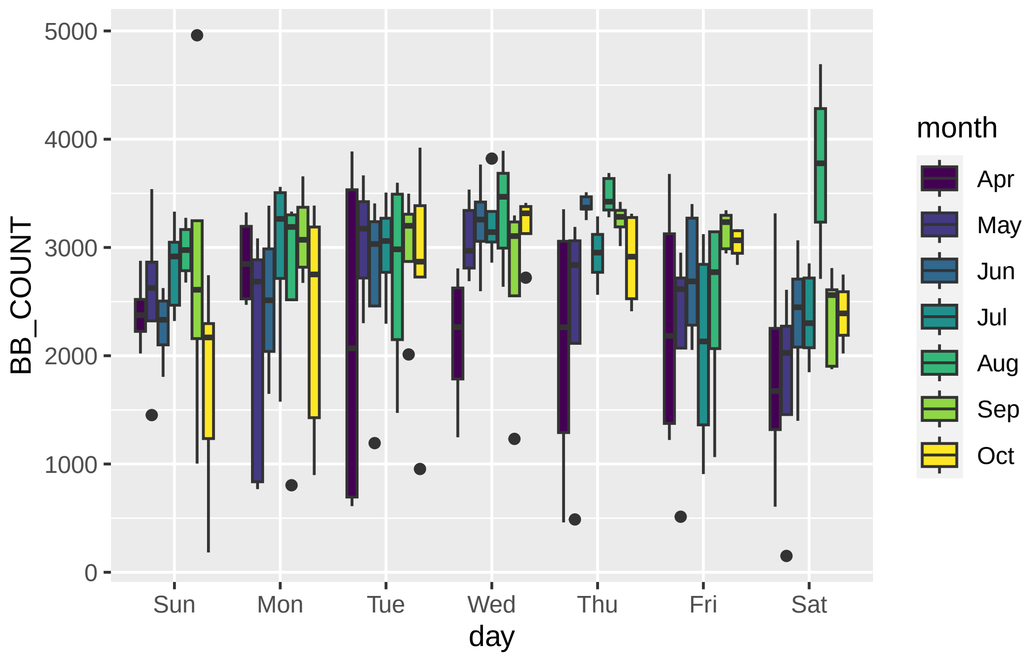
```
brook |>
  ggplot() +
  geom_point(aes(x=HIGH_T, y=LOW_T,
                 size = BB_COUNT,
                 color=BB_COUNT))
```



```
brook |>
  ggplot(aes(x = Date, y = BB_COUNT)) +
  geom_line()
```



```r
brook |>
  ggplot() +
  geom_boxplot(aes(x = day, y = BB_COUNT, fill = month))
```



Modeling

Poisson

First, we'll fit the poisson model. We compare the null model against a model with every covariate, and since we only have a handful of covariates we'll use them all in the model. Here we're basing our model selection on deviance. However, we won't spend too much time fine-tuning our Poisson model since we're going to throw it out anyway based on the poor fit (that I've looked ahead and seen based on the mean-variance relationship).

We see that the extra covariates provide a better fit than the null model, so we will proceed with `pois_mod2`.

```
# null model
pois_mod = glm(BB_COUNT ~ 1,
               family="poisson",
               data = brook)

# lowers deviance, improvement
pois_mod2 = glm(BB_COUNT ~ HIGH_T + LOW_T + PRECIP + num_day + num_month,
               family="poisson",
               data = brook)

anova(pois_mod, pois_mod2)
```

```
Analysis of Deviance Table

Model 1: BB_COUNT ~ 1
Model 2: BB_COUNT ~ HIGH_T + LOW_T + PRECIP + num_day + num_month
  Resid. Df Resid. Dev Df Deviance
1       213      70021
2       198      26666 15    43356
```

Poisson Model Interpretation

Note that our final model is `pois_mod2`.

The point estimates, SE's, and intervals for the model are presented here:

```
coefs = cbind(Estimate = coef(pois_mod2),
              SE = sqrt(diag(vcov(pois_mod2))),
              confint(pois_mod2))
```

```
Waiting for profiling to be done...
```

```
coefs |> kable()
```

	Estimate	SE	2.5 %	97.5 %
(Intercept)	7.0402733	0.0127676	7.0152452	7.0652935
HIGH_T	0.0250786	0.0003050	0.0244809	0.0256764
LOW_T	-0.0190415	0.0003705	-0.0197678	-0.0183153
PRECIP	-0.7658892	0.0069096	-0.7794683	-0.7523813
num_day2	0.1147461	0.0050550	0.1048387	0.1246542
num_day3	0.1608275	0.0050097	0.1510094	0.1706474
num_day4	0.1507624	0.0049222	0.1411162	0.1604110
num_day5	0.1696292	0.0049757	0.1598781	0.1793824
num_day6	0.0951671	0.0051070	0.0851576	0.1051768
num_day7	0.0417767	0.0051922	0.0315997	0.0519529
num_month5	0.0911704	0.0054407	0.0805078	0.1018350
num_month6	0.1446065	0.0060762	0.1326987	0.1565170
num_month7	0.1274220	0.0066708	0.1143499	0.1404990
num_month8	0.2159082	0.0061191	0.2039179	0.2279043
num_month9	0.1488699	0.0058195	0.1374659	0.1602779
num_month10	0.1030984	0.0054099	0.0924964	0.1137030

For interpretation, we exponentiate the coefficients. Note that we have `num_day1` and `num_month4` as reference groups which correspond to Sunday and April respectively. Now, we can say the following:

1. $exp\{\hat{\beta}_{HIGH\_T}\}$: A one unit increase in the high temp results in an expected multiplicative increase of rider count of 1.025 (and a similar interpretation for LOW_T and PRECIP). So precipitation and lower temperatures negatively affect rider counts, as expected.
2. $exp\{\hat{\beta}_{num\_day2}\}$: With Sunday (`num_day1`) as the baseline,we say that on Monday (`num_day2`), on average we expect a ridership increase of 1.12 times compared to Sunday. The same reasoning applies for the rest of the days.
3. $exp\{\hat{\beta}_{num\_month5}\}$: Similar reasoning applies again, but with April (`num_month4`) as the baseline. Thus, in May (`num_month5`), we expect to see an increase vs. April of 1.095 times as many riders. The same holds for the rest of the months.
4. $exp\{\hat{\beta}_0\}$: The intercept represents the expected rider counts on Sunday in April with an expected high and low of 0 and no precipitation.

```
exp(coefs) |> kable()
```

	Estimate	SE	2.5 %	97.5 %
(Intercept)	1141.6996114	1.012849	1113.4796767	1170.6255463
HIGH_T	1.0253957	1.000305	1.0247830	1.0260088
LOW_T	0.9811386	1.000371	0.9804263	0.9818514
PRECIP	0.4649203	1.006934	0.4586498	0.4712430
num_day2	1.1215886	1.005068	1.1105315	1.1327567
num_day3	1.1744824	1.005022	1.1630076	1.1860724
num_day4	1.1627204	1.004934	1.1515585	1.1739932
num_day5	1.1848654	1.004988	1.1733678	1.1964782
num_day6	1.0998427	1.005120	1.0888886	1.1109070
num_day7	1.0426616	1.005206	1.0321042	1.0533261
num_month5	1.0954557	1.005456	1.0838373	1.1072008
num_month6	1.1555847	1.006095	1.1419059	1.1694307
num_month7	1.1358963	1.006693	1.1211444	1.1508480
num_month8	1.2409884	1.006138	1.2261974	1.2559651
num_month9	1.1605220	1.005837	1.1473626	1.1738371
num_month10	1.1086005	1.005425	1.0969092	1.1204193

Poisson Model Assessment

If the Poisson model fits well, we should have:

$$\frac{D(y, \hat{\mu})}{n-p} \approx \frac{\chi^2}{n-p} \approx 1$$

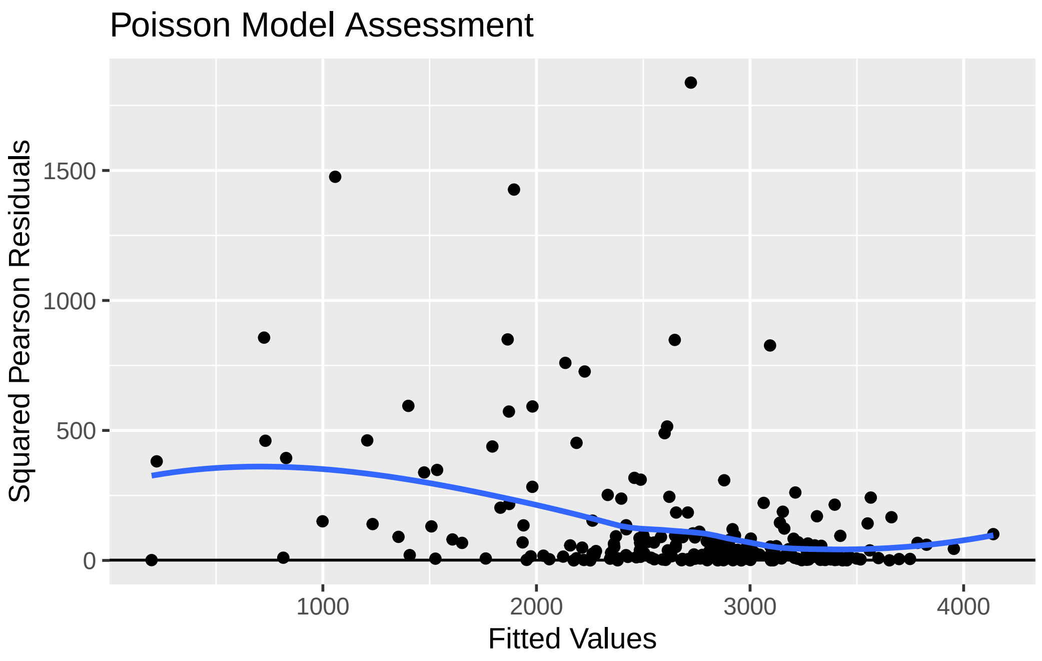
And since our example has values of about 130, the model does not fit well.

Additionally, we can visualize these residuals and fit a smooth curve. If the line is relatively straight around 1, we'll say the model fits well. However, as seen in the plot below, the model fits somewhat well for large fitted values, but we have huge residuals at smaller fitted values, so we'll conclude that the Poisson model fits poorly.

```r
sq_pearson_resid = residuals(pois_mod2, type = 'pearson')^2
resid_df = data.frame(resid = sq_pearson_resid,
                      fitted = pois_mod2$fitted.values)

# plotting
resid_df |>
  ggplot(aes(x=fitted, y=resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_hline(yintercept = 1) +
  labs(
    title = "Poisson Model Assessment",
    x = "Fitted Values",
    y = "Squared Pearson Residuals"
  )
```

```
`geom_smooth()` using formula = 'y ~ x'
```



```
# should be close to 1
pois_mod2$deviance / pois_mod2$df.residual
```

```
[1] 134.6744
```

```
# should be close to 1
sum(sq_pearson_resid) / pois_mod2$df.residual
```

```
[1] 133.3526
```

Negative Binomial

Here we fit the model using the exact same covariates, but using a Negative Binomial distribution on the outcome counts instead of a Poisson to try to fix the overdispersion issue.

Note that the final model is `nbin_m1`

```
nbin_m1 = glm.nb(BB_COUNT ~ HIGH_T + LOW_T + PRECIP + num_day + num_month,
               data = brook)
```

Negative Binomial Model Assessment

Using a likelihood ratio test, we reject the null that the more complex model (Negative Binomial) doesn't offer improvement over the null model (the Poisson Model). Additionally, we fit the same plot we did earlier with a smooth curve and notice that the errors are smaller, though the model still does not fit perfectly.

```
# LRT
lmtest::lrtest(pois_mod2, nbin_m1)
```

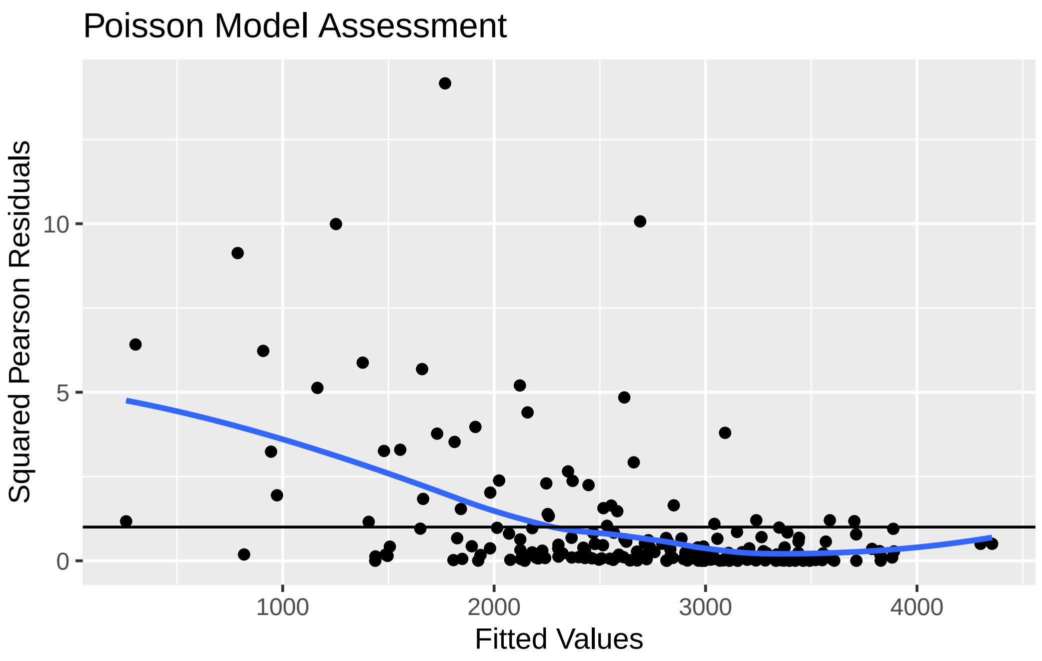
```
Likelihood ratio test

Model 1: BB_COUNT ~ HIGH_T + LOW_T + PRECIP + num_day + num_month
Model 2: BB_COUNT ~ HIGH_T + LOW_T + PRECIP + num_day + num_month
  #Df   LogLik Df Chisq Pr(>Chisq)
1  16 -14364.8
2  17  -1694.2  1 25341  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
sq_pearson_resid = residuals(nbin_m1, type = 'pearson')^2
resid_df = data.frame(resid = sq_pearson_resid,
                      fitted = nbin_m1$fitted.values)

# plotting
resid_df |>
  ggplot(aes(x=fitted, y=resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_hline(yintercept = 1) +
  labs(
    title = "Poisson Model Assessment",
    x = "Fitted Values",
    y = "Squared Pearson Residuals"
  )
```

```
`geom_smooth()` using formula = 'y ~ x'
```



We have the same interpretation of the (exponentiated) coefficients here that we had in the Poisson model. Recall that we have `num_day1` and `num_month4` as reference groups which correspond to Sunday and April respectively.

1. $exp\{\hat{\beta}_{HIGH\_T}\}$: A one unit increase in the high temp results in an expected multiplicative increase of rider count of 1.032 (and a similar interpretation for LOW_T and PRECIP). So precipitation and lower temperatures negatively affect rider counts, as expected. Though temperature and precipitation are on different scales, so a one-unit increase in precipitation is a lot of rain (probably in inches) while a unit increase in temperature is fairly mild.

2. $exp\{\hat{\beta}_{num\_day2}\}$: With Sunday (`num_day1`) as the baseline,we say that on Monday (`num_day2`), on average we expect a ridership increase of 1.102 times compared to Sunday. The same reasoning applies for the rest of the day compared to Sunday as a reference level.

3. $exp\{\hat{\beta}_{num\_month5}\}$: Similar reasoning applies again, but with April (`num_month4`) as the baseline. Thus, in May (`num_month5`), we expect to see an increase vs. April of 1.095 times as many riders. The same holds for the rest of the months compared to the April baseline.

4. $exp\{\hat{\beta}_0\}$: The intercept represents the expected rider counts on Sunday in April with an expected high and low of 0 and no precipitation.

```
coefs = cbind(Estimate = coef(nbin_m1),
              SE = sqrt(diag(vcov(nbin_m1))),
              suppressMessages(confint(nbin_m1)))

exp(coefs) |> kable()
```

	Estimate	SE	2.5 %	97.5 %
(Intercept)	881.9174179	1.185427	624.6836884	1246.3985016
HIGH_T	1.0323320	1.004092	1.0238774	1.0408572
LOW_T	0.9767206	1.005039	0.9669192	0.9866211
PRECIP	0.5205033	1.051070	0.4757576	0.5718205
num_day2	1.1026062	1.070075	0.9657090	1.2589104
num_day3	1.1642303	1.070533	1.0190070	1.3301396
num_day4	1.1644348	1.070718	1.0185820	1.3312946
num_day5	1.1783032	1.071164	1.0298006	1.3483283
num_day6	1.1263936	1.070782	0.9841468	1.2893687
num_day7	1.0407181	1.070158	0.9107219	1.1892732
num_month5	1.1452130	1.072688	0.9982202	1.3137482
num_month6	1.2088668	1.085742	1.0302742	1.4183676
num_month7	1.1736374	1.094505	0.9852919	1.3972880
num_month8	1.2619124	1.088130	1.0704562	1.4869494
num_month9	1.1818049	1.082103	1.0147267	1.3762624
num_month10	1.1347263	1.074034	0.9885319	1.3024731

Limitations

Since the Negative Binomial model still doesn't fit perfectly, we can try a quasi-Poisson model. Because we have overdispersion, the quasi-Poisson model fits better than the original poisson model. While the point estimates are identical, the quasi model has wider confidence intervals.

We could probably reasonably choose either the NBIN model or quasi-poisson. Our analyses might be limited by lack of additional covariates that could help us fit better models, like air quality measures, whether or not there were events around the bridge that day, humidity measures, and other useful pieces of information.

```
quasi_poisson = glm(BB_COUNT ~ HIGH_T + LOW_T + PRECIP + num_day+  num_month,
                    family="quasipoisson",
                    data = brook)

pois_table = data.frame(cbind(std_estimate = coef(pois_mod2),
                              SE = sqrt(diag(vcov(pois_mod2))),
                              suppressMessages(confint(pois_mod2))))

quasi_table = data.frame(cbind(quasi_estimate = coef(quasi_poisson),
                               SE = sqrt(diag(vcov(quasi_poisson))),
                               suppressMessages(confint(quasi_poisson))))

colnames(pois_table) = c("poisson_coef", "poisson_SE", "poisson_2.5%",
                         "poisson_97.5%")
colnames(quasi_table) = c("quasi_coef","quasi_SE", "quasi_2.5%", "quasi_97.5%")

pois_table |> kable()
```

	poisson_coef	poisson_SE	poisson_2.5%	poisson_97.5%
(Intercept)	7.0402733	0.0127676	7.0152452	7.0652935
HIGH_T	0.0250786	0.0003050	0.0244809	0.0256764
LOW_T	-0.0190415	0.0003705	-0.0197678	-0.0183153
PRECIP	-0.7658892	0.0069096	-0.7794683	-0.7523813
num_day2	0.1147461	0.0050550	0.1048387	0.1246542
num_day3	0.1608275	0.0050097	0.1510094	0.1706474
num_day4	0.1507624	0.0049222	0.1411162	0.1604110
num_day5	0.1696292	0.0049757	0.1598781	0.1793824
num_day6	0.0951671	0.0051070	0.0851576	0.1051768
num_day7	0.0417767	0.0051922	0.0315997	0.0519529
num_month5	0.0911704	0.0054407	0.0805078	0.1018350
num_month6	0.1446065	0.0060762	0.1326987	0.1565170
num_month7	0.1274220	0.0066708	0.1143499	0.1404990
num_month8	0.2159082	0.0061191	0.2039179	0.2279043
num_month9	0.1488699	0.0058195	0.1374659	0.1602779
num_month10	0.1030984	0.0054099	0.0924964	0.1137030

```
quasi_table |> kable()
```

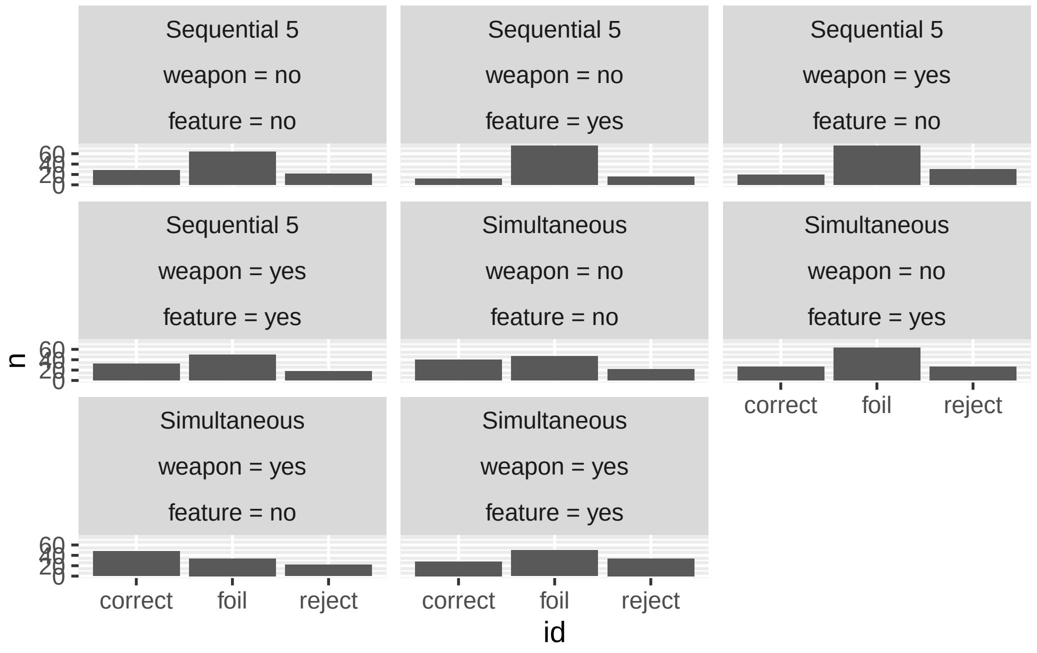
	quasi_coef	quasi_SE	quasi_2.5%	quasi_97.5%
(Intercept)	7.0402733	0.1474406	6.7507622	7.3287358
HIGH_T	0.0250786	0.0035217	0.0181783	0.0319838
LOW_T	-0.0190415	0.0042790	-0.0274309	-0.0106569
PRECIP	-0.7658892	0.0797927	-0.9270809	-0.6141860
num_day2	0.1147461	0.0583755	0.0003557	0.2292448
num_day3	0.1608275	0.0578526	0.0475272	0.2743629
num_day4	0.1507624	0.0568415	0.0394829	0.2623549
num_day5	0.1696292	0.0574590	0.0571292	0.2824211
num_day6	0.0951671	0.0589760	-0.0204464	0.2107987
num_day7	0.0417767	0.0599598	-0.0758301	0.1592738
num_month5	0.0911704	0.0628290	-0.0318744	0.2144776
num_month6	0.1446065	0.0701678	0.0072463	0.2823428
num_month7	0.1274220	0.0770345	-0.0232531	0.2787502
num_month8	0.2159082	0.0706632	0.0777776	0.3548089
num_month9	0.1488699	0.0672035	0.0173982	0.2808775
num_month10	0.1030984	0.0624739	-0.0192035	0.2257532

Q 2)

EDA

From this plot we see that the most common mistake is that the participants selected the "foil", the person placed in the lineup to look like the correct perpetrator but was not. In fact, the only combination of covariates where the participants correctly identified the perpetrator was a simultaneous lineup, with a weapon, and no feature. Otherwise, it seems like a mix of correctly identifying the perpetrator and incorrectly concluding that the perpetrator was in the lineup at all, but primarily we see how often people chose the "foil".

```r
eye |>
  group_by(id, lineup, weapon, feature) |>
  count() |>
  mutate(prop = n / sum(n),
         weapon = if_else(weapon == 'yes', 'weapon = yes', 'weapon = no'),
         feature = if_else(feature == 'yes', 'feature = yes',
                           'feature = no')) |>
  ggplot(aes(x = id, y= n)) +
  geom_bar(stat = 'identity') +
  facet_wrap(vars(lineup, weapon, feature))
```



Nominal Data : Multinomial Model

Since we only have a few covariates, we'll use them all. In the nominal setting, we're ignoring any potential ordering of the outcome.

First we'll set up some functions from the lecture notes that we'll use later.

```
# Borrowing some code from the lecture on categorical data, we can use this
# to assess model fit.

summ.MNfit <- function(fit, digits=3)
{
  s <- summary(fit)
  for(i in 2:length(fit$lev))
  {
    ##
    cat("\nLevel", fit$lev[i], "vs. Level", fit$lev[1], "\n")
    ##
    betaHat <- s$coefficients[(i-1),]
    se <- s$standard.errors[(i-1),]
    zStat <- betaHat / se
    pval <- 2 * pnorm(abs(zStat), lower.tail=FALSE)
    ##
    RRR <- exp(betaHat)
    RRR.lo <- exp(betaHat - qnorm(0.975)*se)
    RRR.up <- exp(betaHat + qnorm(0.975)*se)
    ##
    results <- cbind(betaHat, se, pval, RRR, RRR.lo, RRR.up)
    print(round(results, digits=digits))
  }
}

lrtMN = function(fit1, fit2)
  {
  stat <- abs(deviance(fit1)-deviance(fit2))
  residDF <- abs(fit1$edf-fit2$edf)
  return(pchisq(stat, residDF, lower=FALSE))
}
```

Multinomial Model Assessment

We can also check for interaction terms to see whether or not these help fit the data better. So here we just fit the standard additive model, then a model with a weapon:lineup interaction, and one with a feature:weapon interaction.

```
# fit models
m1 = multinom(id ~ feature + weapon + lineup, data = eye)
```

```
# weights:  15 (8 variable)
initial  value 973.370488
iter  10 value 887.181001
final  value 886.728791
converged
```

```
m2 = multinom(id ~ feature + weapon * lineup, data = eye)
```

```
# weights:  18 (10 variable)
initial  value 973.370488
iter  10 value 887.248189
final  value 886.706448
converged
```

```
m3 = multinom(id ~ feature * weapon + lineup, data = eye)
```

```
# weights:  18 (10 variable)
initial  value 973.370488
iter  10 value 887.527161
final  value 884.169404
converged
```

We can do an LRT (using the functions from class) to see which models offer improvement over previous models. Based on the p-values, model 3 with an interaction term for feature:weapon is our best model.

```
lrtMN(m1, m2)
```

```
[1] 0.9779049
```

```
lrtMN(m2, m3)
```

```
[1] 0
```

With a basic `summary()`, we see that we're essentially doing 2 separate regressions and tying them together with the multinomial distribution. The reference level here is `id == 'correct'`.

```
summary(m3)
```

```
Call:
multinom(formula = id ~ feature * weapon + lineup, data = eye)

Coefficients:
       (Intercept) featureyes   weaponyes lineupSimultaneous
foil     0.8914197  0.8344774 -0.04059096         -0.8009433
reject  -0.2852696  0.5501798  0.15661462         -0.2643009
       featureyes:weaponyes
foil             -0.7426068
reject           -0.4046933

Std. Errors:
       (Intercept) featureyes weaponyes lineupSimultaneous featureyes:weaponyes
foil     0.1792349  0.2412703 0.2213169          0.1660165            0.3315028
reject   0.2236753  0.2943910 0.2675518          0.1985453            0.3955648

Residual Deviance: 1768.339
AIC: 1788.339
```

To more easily interpret the coefficients and fit of the model, we can output the summary table below.

Note that I'm using the superscript of the outcome level to denote the coefficients below, so in the first table we have β^{foil} and in the second, β^{rej}. And for all of these, the column `RRR` indicates the relative risk ratio for the given level to the baseline, so foil to correct and reject to correct.

1. $exp\{\hat{\beta}^{foil}_{featureyes}\} = 2.304$: The risk of identifying the foil instead of the correct perpetrator is 2.439x higher if the suspect had an identifying feature.
2. $exp\{\hat{\beta}^{foil}_{weaponyes}\} = 0.960$: The risk of identifying the foil instead of the correct perpetrator is 0.96x if the suspect had a weapon vs. not.
3. $exp\{\hat{\beta}^{foil}_{0}\} = 2.439$: The risk of identifying the foil instead of the correct perpetrator with no features, no weapon, and a sequential 5 lineup is 2.439x.

And the rest of the coefficients for the level foil vs. level correct can be interpreted the exact same way. The reject vs. correct coefficient are also interpreted this way, but now we're comparing the risk of rejecting that the perpetrator is in the lineup vs. correctly identifying. For example:

1. $exp\{\hat{\beta}^{rej}_{featureyes}\} = 1.734$: The risk of rejecting that the perpetrator was in the lineup instead of the correctly identifying them is 1.734x higher if the suspect had an identifying feature.
2. $exp\{\hat{\beta}^{rej}_{lineupSim.}\} = 0.768$: The risk of rejecting that the perpetrator was in the lineup instead of the correctly identifying them is 0.768x if the lineup was presented simulataneously instead of a sequential 5.

I've only interpreted a handful of these coefficients, but again, the rest follow the same format.

```
summ.MNfit(m3)
```

```
Level foil vs. Level correct
                     betaHat    se  pval   RRR RRR.lo RRR.up
```

```
(Intercept)             0.891 0.179 0.000 2.439  1.716  3.465
featureyes              0.834 0.241 0.001 2.304  1.436  3.696
weaponyes              -0.041 0.221 0.854 0.960  0.622  1.482
lineupSimultaneous     -0.801 0.166 0.000 0.449  0.324  0.622
featureyes:weaponyes   -0.743 0.332 0.025 0.476  0.248  0.911

Level reject vs. Level correct
                     betaHat    se  pval   RRR RRR.lo RRR.up
(Intercept)            -0.285 0.224 0.202 0.752  0.485  1.165
featureyes              0.550 0.294 0.062 1.734  0.974  3.087
weaponyes               0.157 0.268 0.558 1.170  0.692  1.976
lineupSimultaneous     -0.264 0.199 0.183 0.768  0.520  1.133
featureyes:weaponyes   -0.405 0.396 0.306 0.667  0.307  1.449
```

Ordinal Data : Cumulative Model

Now we'll treat the order of the data as having meaning. That is, correctly identifying the perpetrator > identifying the foil > rejecting that the perpetrator is in the lineup. The rationale is that, of course, being correct is the best outcome. Identifying the foil is worse, because it's still incorrect, but there is someone intentionally trying to trick the participants. And last, not identifying the perpetrator at all is the worst outcome. Note that our levels are `0 = correct, 1 = foil, 2 = reject`. It is maybe questionable whether or not "foil" is actually better than "reject", and would likely depend on domain-specific knowledge. In a real setting, the risk would be a higher chance of false imprisonment if a person was incorrectly identified, which we want to avoid.

I chose the cumulative logit model because, as discussed in class, it is the most widely used and has a relatively straight forward interpretation. We will choose our final model as the interaction model, which is `id ~ feature * weapon + lineup`. This is the same as we chose previously, so we're using the same covariates so we can compare the two. Overall, though, it seems we could reasonably choose either the nominal or ordinal model.

```r
ord_model = vglm(cbind(Y0, Y1, Y2) ~ feature * weapon + lineup,
                 cumulative(parallel=FALSE, reverse=FALSE),
                 data = eye)
summary(ord_model)
```

```
Call:
vglm(formula = cbind(Y0, Y1, Y2) ~ feature * weapon + lineup,
    family = cumulative(parallel = FALSE, reverse = FALSE), data = eye)
```

```
Coefficients:
                       Estimate Std. Error z value Pr(>|z|)    
(Intercept):1          -1.15829    0.17029  -6.802 1.03e-11 ***
(Intercept):2           1.54131    0.18949   8.134 4.16e-16 ***
featureyes:1           -0.73618    0.22998  -3.201  0.00137 ** 
featureyes:2            0.01923    0.23923   0.080  0.93594    
weaponyes:1            -0.01672    0.20661  -0.081  0.93549    
weaponyes:2            -0.21789    0.22968  -0.949  0.34280    
lineupSimultaneous:1    0.62685    0.15630   4.011 6.06e-05 ***
lineupSimultaneous:2   -0.26259    0.16445  -1.597  0.11031    
featureyes:weaponyes:1  0.63206    0.31203   2.026  0.04280 *  
featureyes:weaponyes:2 -0.04345    0.32780  -0.133  0.89454    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Names of linear predictors: logitlink(P[Y<=1]), logitlink(P[Y<=2])

Residual deviance: 1769.047 on 1762 degrees of freedom

Log-likelihood: -884.5234 on 1762 degrees of freedom

Number of Fisher scoring iterations: 5 

No Hauck-Donner effect found in any of the estimates


Exponentiated coefficients:
          featureyes:1           featureyes:2            weaponyes:1 
             0.4789387              1.0194127              0.9834171 
           weaponyes:2   lineupSimultaneous:1   lineupSimultaneous:2 
             0.8042144              1.8717016              0.7690552 
featureyes:weaponyes:1 featureyes:weaponyes:2 
             1.8814789              0.9574767 
```

As usual, we exponentiate the coefficients for easier interpretation:

1.$exp\{\hat{\beta}_0^1\} = 0.314$: The odds ratio of identifying the foil or correctly identifying the perpetrator for suspects with no identifying feature, no weapon, and in a simultaneous 5 lineup, is 0.314.

2.$exp\{\hat{\beta}_0^2\} = 4.671$: The odds ratio of identifying the foil or correctly identifying the perpetrator, or incorrectly rejecting that the perpetrator was in the lineup for suspects with no identifying feature, no weapon, and in a simultaneous 5 lineup, is 0.314.

3.$exp\{\hat{\beta}_{featureyes:1}\} = 0.479$: The odds ratio of identifying the foil or correctly identifying the perpetrator for suspects that had an identifying feature, with all other covariates held constant, is 0.479. This is also the cumulative odds ratio of levels 0 and 1.

4.$exp\{\hat{\beta}_{featureyes:2}\} = 1.019$: The odds ratio of identifying the incorrect suspect or the foil or correctly identifying the perpetrator for suspects that had an identifying feature, with all other covariates held constant, is 1.019. This is also the cumulative odds ratio of levels 0, 1, and 2.

The rest of the covariates follow a similar interpretation based on their level as well.

```r
coefs = cbind(Estimate = coef(ord_model),
              SE = sqrt(diag(vcov(ord_model))),
              suppressMessages(confint(ord_model)))

exp(coefs) |> kable()
```

	Estimate	SE	2.5 %	97.5 %
(Intercept):1	0.3140235	1.185651	0.2249103	0.4384449
(Intercept):2	4.6706853	1.208634	3.2217069	6.7713488
featureyes:1	0.4789387	1.258571	0.3051570	0.7516861
featureyes:2	1.0194127	1.270266	0.6378526	1.6292201
weaponyes:1	0.9834171	1.229500	0.6559535	1.4743563
weaponyes:2	0.8042144	1.258200	0.5127030	1.2614726
lineupSimultaneous:1	1.8717016	1.169175	1.3778278	2.5426015
lineupSimultaneous:2	0.7690552	1.178745	0.5571565	1.0615436
featureyes:weaponyes:1	1.8814789	1.366189	1.0207116	3.4681324
featureyes:weaponyes:2	0.9574767	1.387917	0.5036173	1.8203538