HW4

Isaac Kleisle-Murphy

3/4/2022

11.3

a.)

```
data = matrix(
  c(955, 162, 9, 188),
  byrow = TRUE, ncol = 2
) %>%
  data.frame() %>%
  `colnames<-`(c("yes_hell", "no_hell")) %>%
  `rownames<-`(c("yes_heav", "no_heav"))

data
```

```
##          yes_hell no_hell
## yes_heav      955     162
## no_heav         9     188
```

As set forth in Agresti 11.1, we compute statistic (using marginal proportions) $d = \hat{p}_{\cdot,1} - \hat{p}_{1,\cdot}$, as well as (take the corners) $\hat{\sigma}_d^2 = \frac{\hat{p}_{1,2} - \hat{p}_{2,1} - (\hat{p}_{1,2} + \hat{p}_{2,1})^2}{\sum p_{ij}}$, and then we get a Wald CI with d as mean and $\hat{\sigma}_d^2$ as variance. Coded up, this is:

```
p1_ = (rowSums(data) / sum(data))[1]
p_1 = (colSums(data) / sum(data))[1]
d = p1_ - p_1

p12 = data[1, 2] / sum(data); p21 = data[2, 1] / sum(data)
sigma2_d = (p12 + p21 - (p12 - p21)^2) / sum(data)

cat("d =", d, "\nsigma2 =", sigma2_d)
```

```
## d = 0.1164384
## sigma2 = 8.872077e-05
```

And hence a Wald CI of:

```
d - qnorm(.975) * sqrt(sigma2_d); d + qnorm(.975) * sqrt(sigma2_d)
```

```
##   yes_heav
## 0.09797712
```

```
##  yes_heav
## 0.1348996
```

We then say that we are 95% confident that the true difference in marginal proportions between those who believe in heaven and those who believe in hell is between .098 and .135.

b.)

As detailed in Agresti 11.1.2, our test statistic under null is $T = \frac{d}{\hat{\sigma}_0^{(d)}} = \frac{n_{21}-n_{12}}{(n_{21}+n_{12})^{1/2}}$, where our null is that the marginals are equal ($H0 : d = 0 \iff \pi_{1,\cdot} = \pi_{\cdot,1}$):

```
(162 - 9) / sqrt(162 + 9)
```

```
## [1] 11.7002
```

We then square this statistic, and test T^2 against a χ^2_1. The p-value is zero, leading to a rejection:

```
pchisq(11.7 * 11.7, 1, lower.tail=F)
```

```
## [1] 1.274535e-31
```

Hence, we reject the marginal homogeneity assumption – clearly there is a significant difference in the marginal proportions, for a "yes" answer belief in belief in heaven and belief in hell.

c.)

As set forth in Agresti 11.1, we have that

$$VAR(\sqrt{n}d) = \pi_{1,\cdot}(1-\pi_{1,\cdot}) + \pi_{\cdot,1}(1-\pi_{\cdot,1}) - 2(\prod_{i=j}\pi_{ij} - \prod_{i\neq j}\pi_{ij}) = VAR(\pi_{1,\cdot}) + VAR(\pi_{\cdot,1}) - 2COV(\pi_{1,\cdot}, \pi_{\cdot,1})$$

And if the covariance term is nonzero and large, we know there is dependence between the two. Hence, when we see this massive value:

```
data[2, 2] * data[1, 1] - data[1, 2] * data[2, 1]
```

```
## [1] 178082
```

we can be pretty confident in a high covariance, and hence dependence.

Now, had we taken independent samples, then we'd have $2COV(\pi_{1,\cdot}, \pi_{\cdot,1}) = 0$, due to the independence of our procedure. Accordingly, this covariance would not be baked into our variance calculation, and any corresponding test would overlook the fundamental dependence here. This could induce a misleading test result.

11.9

a.)

Recall the symmetric setup, i.e.

$$\log(\mu_{ab}) = \lambda + \lambda_a + \lambda_b + \underbrace{\lambda_{ab}}_{=\lambda_{ba}}$$

We then wrangle:

```
data_base = data.frame(first_purchase=c("high_point", "tasters_choice", "sanka", "nescafe", "brim"),
  high_point=c(93, 9, 17, 6, 10),
  tasters_choice=c(17, 46, 11, 4, 4),
  sanka=c(44, 11, 155, 9, 12),
  nescafe=c(7, 0, 9, 15, 2),
  brim=c(10, 9, 12, 2, 27)
)

data =  data_base %>%
  pivot_longer(
    high_point:brim, names_to="second_purchase", values_to="count",
  ) %>%
  mutate(
    sym_pairing = ifelse(
      first_purchase < second_purchase,
      paste0(first_purchase, "-",  second_purchase),
      paste0(second_purchase, "-",  first_purchase)
    )
  )
data
```

```
## # A tibble: 25 x 4
##    first_purchase second_purchase count sym_pairing
##    <chr>          <chr>           <dbl> <chr>
##  1 high_point     high_point         93 high_point-high_point
##  2 high_point     tasters_choice     17 high_point-tasters_choice
##  3 high_point     sanka              44 high_point-sanka
##  4 high_point     nescafe             7 high_point-nescafe
##  5 high_point     brim               10 brim-high_point
##  6 tasters_choice high_point          9 high_point-tasters_choice
##  7 tasters_choice tasters_choice     46 tasters_choice-tasters_choice
##  8 tasters_choice sanka              11 sanka-tasters_choice
##  9 tasters_choice nescafe             0 nescafe-tasters_choice
## 10 tasters_choice brim                9 brim-tasters_choice
## # ... with 15 more rows
```

and fit the symmetry model:

```
glm(count ~ sym_pairing, data, family=poisson()) %>%
  summary()
```

```
##
## Call:
```

```
## glm(formula = count ~ sym_pairing, family = poisson(), data = data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.670   0.000   0.000   0.000   2.291
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                            3.2958     0.1925  17.126  < 2e-16
## sym_pairingbrim-high_point            -0.9933     0.2950  -3.367 0.000761
## sym_pairingbrim-nescafe               -2.6027     0.5358  -4.858 1.19e-06
## sym_pairingbrim-sanka                 -0.8109     0.2805  -2.891 0.003845
## sym_pairingbrim-tasters_choice        -1.4240     0.3376  -4.218 2.46e-05
## sym_pairinghigh_point-high_point       1.2368     0.2186   5.657 1.54e-08
## sym_pairinghigh_point-nescafe         -1.4240     0.3376  -4.218 2.46e-05
## sym_pairinghigh_point-sanka            0.1219     0.2312   0.527 0.597973
## sym_pairinghigh_point-tasters_choice  -0.7309     0.2748  -2.660 0.007814
## sym_pairingnescafe-nescafe            -0.5878     0.3220  -1.825 0.067963
## sym_pairingnescafe-sanka              -1.0986     0.3043  -3.610 0.000306
## sym_pairingnescafe-tasters_choice     -2.6027     0.5357  -4.858 1.19e-06
## sym_pairingsanka-sanka                 1.7476     0.2085   8.380  < 2e-16
## sym_pairingsanka-tasters_choice       -0.8979     0.2872  -3.126 0.001770
## sym_pairingtasters_choice-tasters_choice  0.5328     0.2424   2.198 0.027971
##
## (Intercept)                           ***
## sym_pairingbrim-high_point            ***
## sym_pairingbrim-nescafe               ***
## sym_pairingbrim-sanka                 **
## sym_pairingbrim-tasters_choice        ***
## sym_pairinghigh_point-high_point      ***
## sym_pairinghigh_point-nescafe         ***
## sym_pairinghigh_point-sanka
## sym_pairinghigh_point-tasters_choice  **
## sym_pairingnescafe-nescafe            .
## sym_pairingnescafe-sanka              ***
## sym_pairingnescafe-tasters_choice     ***
## sym_pairingsanka-sanka                ***
## sym_pairingsanka-tasters_choice       **
## sym_pairingtasters_choice-tasters_choice *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 768.544  on 24  degrees of freedom
## Residual deviance:  22.473  on 10  degrees of freedom
## AIC: 157.01
##
## Number of Fisher Scoring iterations: 5
```

We see a residual deviance from an empty Poisson regression of 22.47; on 10 df, this is concerning, as we'd want residual deviance roughly equal to degrees of freedom. Our fit here appears to be poor. If we say H_0 is symmetry, and test using this statistic, we get a p-value of

```
1 - pchisq(22.47, 10)
```

```
## [1] 0.01288132
```

which likely results in rejection of the nullmodel, i.e. symmetry. On inspection, it's clear that much of this rejection is powered by the Sanka/high-point pairing, which has heavily imbalanced corners (17 and 44):

```
data_base
```

```
##   first_purchase high_point tasters_choice sanka nescafe brim
## 1     high_point         93             17    44       7   10
## 2 tasters_choice          9             46    11       0    9
## 3          sanka         17             11   155       9   12
## 4        nescafe          6              4     9      15    2
## 5           brim         10              4    12       2   27
```

This `high_point` vs. `sanka` imbalance causes `(44 - 17) / sqrt(44 + 17) ~= 3.5`, or over 9 points to the X^2. This is the bulk of the differential between X^2 (analogously, G^2) and `df`.

b.)

The marginal homogeneity test, as set forth in Agresti 11.25, compares the symmetry model with the quasi-symmetry model. So we need to fit the quasi-symmetry model, which recall takes the form

$$\log(\mu_{ab}) = \lambda + \lambda_a + \lambda_b + \lambda_{ab},$$

but now $\lambda_{ab} = \lambda_{ba}$ for $a < b$

Fit it:

```
glm(count ~ sym_pairing + first_purchase, data=data, family=poisson()) %>%
  summary()
```

```
##
## Call:
## glm(formula = count ~ sym_pairing + first_purchase, family = poisson(),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8419  -0.1925   0.0000   0.1999   1.0215
##
## Coefficients:
##                                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)                            3.295837   0.192450  17.126  < 2e-16
## sym_pairingbrim-high_point            -1.334651   0.350528  -3.808 0.000140
## sym_pairingbrim-nescafe               -2.765113   0.583396  -4.740 2.14e-06
## sym_pairingbrim-sanka                 -0.755884   0.311106  -2.430 0.015113
## sym_pairingbrim-tasters_choice        -1.422034   0.375461  -3.787 0.000152
## sym_pairinghigh_point-high_point       0.641323   0.366094   1.752 0.079808
## sym_pairinghigh_point-nescafe         -1.883529   0.449603  -4.189 2.80e-05
## sym_pairinghigh_point-sanka           -0.180698   0.356920  -0.506 0.612668
```

```
## sym_pairinghigh_point-tasters_choice     -1.070865   0.389101  -2.752 0.005921
## sym_pairingnescafe-nescafe               -0.889902   0.514759  -1.729 0.083850
## sym_pairingnescafe-sanka                 -1.214438   0.440553  -2.757 0.005840
## sym_pairingnescafe-tasters_choice        -2.763413   0.622295  -4.441 8.97e-06
## sym_pairingsanka-sanka                    1.860887   0.353214   5.268 1.38e-07
## sym_pairingsanka-tasters_choice          -0.840782   0.397821  -2.113 0.034561
## sym_pairingtasters_choice-tasters_choice  0.536809   0.408966   1.313 0.189318
## first_purchasehigh_point                  0.595440   0.293658   2.028 0.042595
## first_purchasenescafe                     0.302115   0.401589   0.752 0.451871
## first_purchasesanka                      -0.113299   0.285082  -0.397 0.691052
## first_purchasetasters_choice             -0.004004   0.329359  -0.012 0.990299
##
## (Intercept)                              ***
## sym_pairingbrim-high_point               ***
## sym_pairingbrim-nescafe                  ***
## sym_pairingbrim-sanka                    *
## sym_pairingbrim-tasters_choice           ***
## sym_pairinghigh_point-high_point         .
## sym_pairinghigh_point-nescafe            ***
## sym_pairinghigh_point-sanka
## sym_pairinghigh_point-tasters_choice     **
## sym_pairingnescafe-nescafe               .
## sym_pairingnescafe-sanka                 **
## sym_pairingnescafe-tasters_choice        ***
## sym_pairingsanka-sanka                   ***
## sym_pairingsanka-tasters_choice          *
## sym_pairingtasters_choice-tasters_choice
## first_purchasehigh_point                 *
## first_purchasenescafe
## first_purchasesanka
## first_purchasetasters_choice
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 768.544  on 24  degrees of freedom
## Residual deviance:   9.974  on  6  degrees of freedom
## AIC: 152.51
##
## Number of Fisher Scoring iterations: 5
```

As instructed by Agresti, we subtract the residual deviance from the symmetry model from that of the quasi-symmetry, i.e.

$$G^2(Sym) - G^2(QuasiSym) = 22.473 - 9.974 = 12.499.$$

The delta in degrees of freedom is 10-6=4 so we test 12.499 against 4 df:

```
1 - pchisq(12.499, 4)
```

```
## [1] 0.01400183
```

Here, our null hypothesis would have been marginal homogeneity; this p-value likely induces (for reasonable $\alpha = .05$) a rejection in favor of marginal non-homogeneity. This small p-value is again overwhelmingly driven

by the High Point / Sanka pairing. If we do a one vs. all collapsing (i.e. high point vs. the field), and take a McNemar's statistic, we have

```
data %>%
  mutate(
    first_purchase = ifelse(first_purchase == "high_point", "high_point", "other"),
    second_purchase = ifelse(second_purchase == "high_point", "high_point", "other")
  ) %>%
  group_by(first_purchase, second_purchase) %>%
  summarise(count_ = sum(count)) -> data_collapsed
```

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

```
data_collapsed
```

```
## # A tibble: 4 x 3
## # Groups:   first_purchase [2]
##   first_purchase second_purchase count_
##   <chr>          <chr>            <dbl>
## 1 high_point     high_point          93
## 2 high_point     other               78
## 3 other          high_point          42
## 4 other          other              328
```

This gives a 2x2 table of (high point in 0 index always; row = first; column = second)

$$\begin{bmatrix} & HP2 & O2 \\ HP1 & 93 & 78 \\ O1 & 42 & 328 \end{bmatrix}$$

The McNemar's stat here is $(78 - 42)/\sqrt{78 + 42} = 3.286$. This value is pretty extreme, showing how **high_point** – in particular, the imbalance between **high_point** and **sanka** (also demonstrated above) – is again driving the rejection.

For the Wald CI, we use the procedure in the previous problem: compute d and $\hat{\sigma}^2$, and back out a CI from this. That is, s set forth in Agresti 11.1, we compute statistic (using marginal proportions) $d = \hat{p}_{\cdot,1} - \hat{p}_{1,\cdot}$, as well as (take the corners) $\hat{\sigma}_d^2 = \frac{\hat{p}_{1,2} - \hat{p}_{2,1} - (\hat{p}_{1,2} + \hat{p}_{2,1})^2}{\sum p_{ij}}$, and then we get a Wald CI with d as mean and $\hat{\sigma}_d^2$ as variance. Coded up, this is:

```
data_ = matrix(c(93, 78, 42, 328), byrow=T, ncol=2)

p1_ = (rowSums(data_) / sum(data_))[1]
p_1 = (colSums(data_) / sum(data_))[1]
d = p_1 - p1_

p12 = data_[1, 2] / sum(data_);
p21 = data_[2, 1] / sum(data_)
sigma2_d = (p12 + p21 - (p12 - p21)^2) / sum(data_)

cat("d =", d, "\nsigma2 =", sigma2_d)
```

```
## d = -0.06654344
## sigma2 = 0.0004018178
```

The Wald CI is thus

```
d - sqrt(sigma2_d) * qnorm(.975); d + sqrt(sigma2_d) * qnorm(.975)
```

```
## [1] -0.1058317
```

```
## [1] -0.02725519
```

That is, we say with about 95% confidence that the true difference in marginal proportions is between around -.105 and -.027.

c.)

First, let's wrangle the diagonal indicators before fitting either

```
data_diag = data %>%
  mutate(diag_indicator = ifelse(
    first_purchase == second_purchase,
    first_purchase,
    "0")
  )

data_diag
```

```
## # A tibble: 25 x 5
##    first_purchase second_purchase count sym_pairing               diag_indicator
##    <chr>          <chr>           <dbl> <chr>                     <chr>
##  1 high_point     high_point         93 high_point-high_point     high_point
##  2 high_point     tasters_choice     17 high_point-tasters_choice 0
##  3 high_point     sanka              44 high_point-sanka          0
##  4 high_point     nescafe             7 high_point-nescafe        0
##  5 high_point     brim               10 brim-high_point           0
##  6 tasters_choice high_point          9 high_point-tasters_choice 0
##  7 tasters_choice tasters_choice     46 tasters_choice-tasters_c~ tasters_choice
##  8 tasters_choice sanka              11 sanka-tasters_choice      0
##  9 tasters_choice nescafe             0 nescafe-tasters_choice    0
## 10 tasters_choice brim                9 brim-tasters_choice       0
## # ... with 15 more rows
```

We then proceed to fit the quasi-independence model, in order to test marginal homogeity

```
glm(count ~ first_purchase + second_purchase + diag_indicator, data=data_diag, family=poisson()) %>%
  summary()
```

```
##
## Call:
## glm(formula = count ~ first_purchase + second_purchase + diag_indicator,
##     family = poisson(), data = data_diag)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.2123  -0.3375   0.0000   0.2768   1.7751
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    1.49011    0.27083   5.502 3.75e-08 ***
## first_purchasehigh_point       1.16100    0.22808   5.090 3.57e-07 ***
## first_purchasenescafe         -0.36039    0.29056  -1.240 0.214847
## first_purchasesanka            0.91381    0.24899   3.670 0.000243 ***
## first_purchasetasters_choice   0.05084    0.26808   0.190 0.849575
## second_purchasehigh_point      0.59060    0.24318   2.429 0.015156 *
## second_purchasenescafe        -0.64602    0.29479  -2.191 0.028417 *
## second_purchasesanka           1.05846    0.21839   4.847 1.26e-06 ***
## second_purchasetasters_choice  0.09405    0.24384   0.386 0.699718
## diag_indicatorbrim             1.80573    0.33224   5.435 5.48e-08 ***
## diag_indicatorhigh_point       1.29089    0.24603   5.247 1.55e-07 ***
## diag_indicatornescafe          2.22435    0.41972   5.300 1.16e-07 ***
## diag_indicatorsanka            1.58105    0.23148   6.830 8.48e-12 ***
## diag_indicatortasters_choice   2.19364    0.30257   7.250 4.17e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 768.544  on 24  degrees of freedom
## Residual deviance:  13.786  on 11  degrees of freedom
## AIC: 146.32
##
## Number of Fisher Scoring iterations: 5
```

Just as anticipated: 13.78 residual deviance on 11 df. With a p-value of

```
1- pchisq(13.786, 11)
```

```
## [1] 0.245062
```

we'd have little reason to reject a null hypothesis of the quasi-independence model. We can live with this quasi-independence model, which implies that when people change brands, it appears their second choice is independent of their first choice.

By contrast, the *independence* model looks like

```
glm(count ~ first_purchase + second_purchase , data=data_diag, family=poisson()) %>%
  summary()
```

```
##
## Call:
## glm(formula = count ~ first_purchase + second_purchase, family = poisson(),
##     data = data_diag)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.5246  -2.5005  -1.1303  -0.6549   7.7031
##
```

```
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      1.8083     0.1817   9.954  < 2e-16 ***
## first_purchasehigh_point         1.1343     0.1550   7.318 2.52e-13 ***
## first_purchasenescafe           -0.4238     0.2144  -1.977   0.0481 *
## first_purchasesanka              1.3108     0.1519   8.627  < 2e-16 ***
## first_purchasetasters_choice     0.3102     0.1775   1.747   0.0806 .
## second_purchasehigh_point        0.8109     0.1552   5.226 1.73e-07 ***
## second_purchasenescafe          -0.5978     0.2167  -2.759   0.0058 **
## second_purchasesanka             1.3481     0.1449   9.304  < 2e-16 ***
## second_purchasetasters_choice    0.3124     0.1699   1.839   0.0660 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 768.54  on 24  degrees of freedom
## Residual deviance: 346.38  on 16  degrees of freedom
## AIC: 468.92
##
## Number of Fisher Scoring iterations: 5
```

346.8 residual deviance on 16 degrees of freedom is awful! This would be a terrible fit, taking p-value

```
1 - pchisq(346.78, 16)
```

```
## [1] 0
```

and a clear-cut rejection of this model.

11.11

```
library(stringr)
#' Recycling code above
data_base = data.frame(
  father_group = c("fat1", "fat2", "fat3", "fat4", "fat5"),
  son1 = c(50, 28, 11, 14, 3),
  son2 = c(45, 174, 78, 150, 42),
  son3 = c(8, 84, 110, 185, 72),
  son4 = c(18, 154, 223, 714, 320),
  son5 = c(8, 55, 96, 447, 411)
)
data_base
```

```
##   father_group son1 son2 son3 son4 son5
## 1         fat1   50   45    8   18    8
## 2         fat2   28  174   84  154   55
## 3         fat3   11   78  110  223   96
## 4         fat4   14  150  185  714  447
## 5         fat5    3   42   72  320  411
```

```r
data =data_base %>%
  pivot_longer(
    son1:son5, names_to="son_group", values_to="count",
  ) %>%
  mutate(
    # for ordinal
    father_group_ord = str_replace_all(father_group, "fat", ""),
    father_group_ord = as.numeric(father_group_ord),
    son_group_ord = str_replace_all(son_group, "son", ""),
    son_group_ord = as.numeric(son_group_ord),
    # symmetry pairings
    sym_pairing = ifelse(
      father_group_ord < son_group_ord,
      paste0(father_group_ord, "-",  son_group_ord),
      paste0(son_group_ord, "-",  father_group_ord)
    ),
    # diagonal indicators
    diag_indicator = ifelse(
      father_group_ord == son_group_ord,
      as.character(father_group_ord),
      "0"
    )
  )

# symmetry
fit_a = glm(count ~ sym_pairing, data=data %>% arrange(son_group), family=poisson())
# quasi symmetry
fit_b = glm(count ~ sym_pairing + father_group, data=data, family=poisson())
# ordinal quasi symmetry
# put feature u = 1, .., 5 in for father group, instead of using the one-hot encoding. That is,
# undiscretize father group
fit_c = glm(count ~ sym_pairing + father_group_ord,
            data=data,
            family=poisson()
)

fit_e = glm(
  count ~ father_group + son_group + diag_indicator,
  data=data,
  family=poisson()
)

### helpers
testmod <- function(mod){1 - pchisq(mod$deviance, mod$df.residual)}
analyze <- function(mod){
  cat("Residual Deviance: ", mod$deviance, "\nResidual DF: ", mod$df.residual)
  cat("\nP-Value: ", testmod(mod))
}
```

a.)

```
analyze(fit_a)
```

```
## Residual Deviance:  37.4637
## Residual DF:  10
## P-Value:  4.703629e-05
```

With 37 deviance on 10 degrees of freedom, this appears to be a poor fit – so poor, a hypothesis test would likely reject it.

b.)

```
analyze(fit_b)
```

```
## Residual Deviance:  4.664071
## Residual DF:  6
## P-Value:  0.5875617
```

Here, residual deviance is in line with residual df; the high p-value returns a no rejection, so this is probably a decent model.

c.)

```
analyze(fit_c)
```

```
## Residual Deviance:  17.12618
## Residual DF:  9
## P-Value:  0.04677634
```

While not as extreme as in a.), residual deviance is well above residual df, and a p-value of .04 rejects at alpha .05. This is probably a poor fit as well.

d.)

Recall we test using delta between symmetric and quasi-symmetric, i.e.

```
cat("S-QS Deviance: ", fit_a$deviance - fit_b$deviance, "\nS-QS DF: ", fit_a$df.residual - fit_b$df.resi
```

```
## S-QS Deviance:  32.79963
## S-QS DF:  4
```

```
cat("\nP-Value: ", 1 - pchisq(fit_a$deviance - fit_b$deviance, fit_a$df.residual - fit_b$df.residual))
```

```
##
## P-Value:  1.312792e-06
```

As before, this too is likely to get rejected if chosen as a null model, given the small p-value here.

e.)

```
analyze(fit_e)
```

```
## Residual Deviance:  235.7818
## Residual DF:  11
## P-Value:  0
```

Same story as before, residual deviance way ahead of residual df. P-value is zero, so this gets rejected as a null.

Given that the quasi-independence model is the only one to not get rejected, it's probably best here.

11.26

Q: Explain the following analogy: McNemar's test is to binary data as the paired difference t test is to normally distributed data.

First, there is an obvious duality between the two: whereas paired t-tests test the difference between two real-valued variables that are dependent on one another, McNemar tests the difference between two marginal proportions/percentages, which are necessarily dependent (in order to fill out the data table). That is, the McNemar's example boils down to the single-voter example for Bush/Gore vs. McCain/Obama: wherever the one voter lands, the rest of the cells are all zero and hence dependent on that first cell.

Moreover, it's also worth noting that generally both begin with an H_0 of zero difference in proportion: for McNemar, this is $\pi_{1,\cdot} - \pi_{\cdot,1} = 0$, whereas for paired it is that $X - Y = 0$.

11.34

Consider the following counter example,

$$P \propto \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{bmatrix}$$

Clearly, this will satisfy marginal homogeneity as for any i, j, we have that (recall that diagonal doesn't count, so we zero it out) $\sum_{j \neq i} p_{i,j} = \sum_{i \neq j} p_{i,j}$. However, $p_{12} \neq p_{2,1}$, for example, so it is non symmetric.

11.32

a.)

For setup and me to keep track of, we have under Agresti's Ch. 11 specification that

- $t = 1, 2$ are the timesteps.
- $s_i = y_{i,t=1} + y_{i,t=2}$ Are the observations
- $Y_{i,t}, S_i$ are the corresponding RVs
- $\beta \in \mathbb{R}^p$ are the coefficients
- $X \in \mathbb{R}^{N \times p \times t}$ is a tensor of features.
- $x_{i,:,t}\mathbb{R}^p$ is a vector of one realization of features at time t.

- The model is then given by, for the $t = 1, 2$ scenario

$$P(Y_{i,t} = 1) = Logit^{-1}(x_{i,:,t}^T \beta).$$

In turn, when looking at the $S_i = 1$ cases, we have

$$\begin{aligned} P(Y_{i,1} = 0, Y_{i,2} = 1 | S_i = 1) &\propto P(Y_{i,0} = 0, Y_{i,1} = 1, S_i = 1) \\ &\propto \underbrace{\left(\frac{1}{1 + exp(x_{i,:,1}^T \beta)} \right)}_{P(Y_{i,1}=0)} \cdot \underbrace{\left(\frac{exp(x_{i,:,2}^T \beta)}{1 + exp(x_{i,:,2}^T \beta)} \right)}_{P(Y_{i,2}=1)} \\ &\propto exp(x_{i,:,2}^T \beta) \\ &= \frac{exp(x_{i,:,2}^T \beta)}{exp(x_{i,:,1}^T \beta) + exp(x_{i,:,2}^T \beta)} \end{aligned}$$

And identically

$$P(Y_{i,1} = 1, Y_{i,2} = 0 | S_i = 1) \propto exp(x_{i,:,1}^T \beta) = \frac{exp(x_{i,:,1}^T \beta)}{exp(x_{i,:,1}^T \beta) + exp(x_{i,:,2}^T \beta)}.$$

The final step above follows from the requisite normalization under conditioning.

In these cases, the dependence on β is clear cut. In contrast, consider what happens when $S_i = 0$. Deterministically, it must be that $Y_{i, 1} = 0, Y_{i, 2} = 0$ under these conditions, and hence

$$P(Y_{i,1} = a, Y_{i,2} = b | S_i = 0) = \begin{cases} 1 & a = b = 0 \\ 0 & else \end{cases}.$$

Similarly, when $S_i = 2$, it has to be that $Y_{i,1} = Y_{i,2} = 1$ (otherwise a contradiction), giving

$$P(Y_{i,1} = a, Y_{i,2} = b | S_i = 2) = \begin{cases} 1 & a = b = 1 \\ 0 & else \end{cases}.$$

Here, due to the deterministic nature of the cases, there is no dependence on β.

b.)

Next, we consider what happens to the corresponding coefficient when one of the features never changes under a timestep. Arbitrarily, say $1 \leq j \leq p$ is that feature, we now care about β_j. Now, when $S_i \in \{0, 2\}$, we know the conditional distribution won't depend on this $\hat{\beta}_j$; we just showed this above. However, when (using $\sim j$ to denote inclusion of all indices except j) $S_i = 1$, we have

$$\begin{aligned} P(Y_{i,1} = 1, Y_{i,2} = 0 | S_i = 1) &= \frac{exp(x_{i,:,1}^T \beta)}{exp(x_{i,:,1}^T \beta) + exp(x_{i,:,2}^T \beta)} \\ &= \frac{exp(x_{i,\sim j,1}^T \beta_{\sim j} + x_{i,j,1} \beta_j)}{exp(x_{i,\sim j,1}^T \beta_{\sim j} + x_{i,j,1} \beta_j) + exp(x_{i,\sim j,2}^T \beta_{\sim j} + x_{i,j,2} \beta_j)} \\ &= \frac{exp(x_{i,\sim j,1}^T \beta_{\sim j}) \exp(x_{i,j,1} \beta_j)}{exp(x_{i,\sim j,1}^T \beta_{\sim j}) \exp(x_{i,j,1} \beta_j) + exp(x_{i,\sim j,2}^T \beta_{\sim j}) \exp(x_{i,j,2} \beta_j)} \\ &= \frac{exp(x_{i,\sim j,1}^T \beta_{\sim j}) \exp(x_{i,j,1} \beta_j)}{exp(x_{i,\sim j,1}^T \beta_{\sim j}) \exp(x_{i,j,1} \beta_j) + exp(x_{i,\sim j,2}^T \beta_{\sim j}) \exp(x_{i,j,1} \beta_j)} \\ &= \frac{exp(x_{i,\sim j,1}^T \beta_{\sim j})}{exp(x_{i,\sim j,1}^T \beta_{\sim j}) + exp(x_{i,\sim j,2}^T \beta_{\sim j})}, \end{aligned}$$

and here, there is no dependence on $\hat{\beta}_j$.

Staphyllococcus

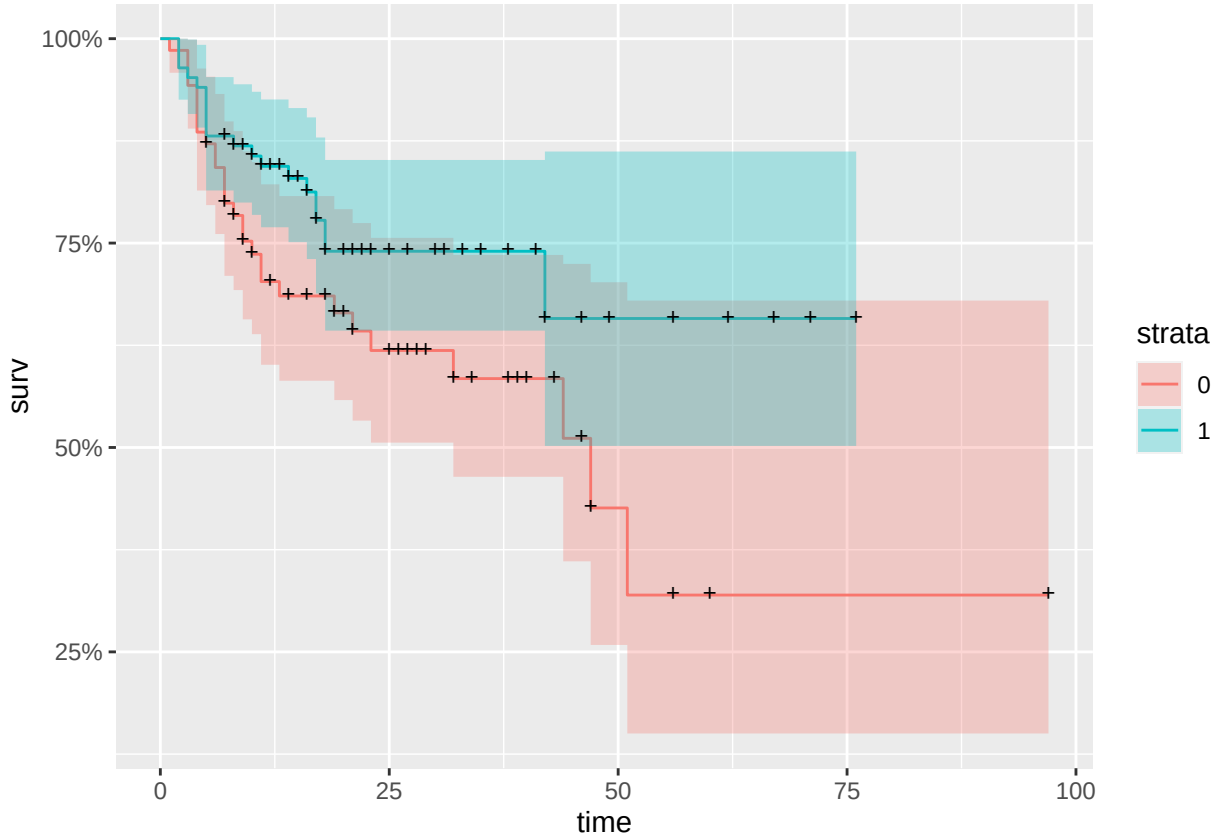
```
head(burn)
```

```
##   Obs Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10 Z11 T1 D1 T2 D2 T3 D3
## 1   1  0  0  0 15  0  0  1  1  0   0   2 12  0 12  0 12  0
## 2   2  0  0  1 20  0  0  1  0  0   0   4  9  0  9  0  9  0
## 3   3  0  0  1 15  0  0  0  1  1   0   2 13  0 13  0  7  1
## 4   4  0  0  0 20  1  0  1  0  0   0   2 11  1 29  0 29  0
## 5   5  0  0  1 70  1  1  1  1  0   0   2 28  1 31  0  4  1
## 6   6  0  0  1 20  1  0  1  0  0   0   4 11  0 11  0  8  1
```

1.)

The stratified K-M plot is presented below:

```
survfit(Surv(T3, D3) ~ Z1, data=burn) -> km_fit
autoplot(km_fit)
```



2.)

Perform the test:

```
survdiff(Surv(T3, D3) ~ Z1, data=burn)
```

```
## Call:
## survdiff(formula = Surv(T3, D3) ~ Z1, data = burn)
##
##       N Observed Expected (O-E)^2/E (O-E)^2/V
## Z1=0 70       28     21.4      2.07      3.79
## Z1=1 84       20     26.6      1.66      3.79
##
##  Chisq= 3.8  on 1 degrees of freedom, p= 0.05
```

It's a really close call, with a p-value of .05. Assuming we're testing at $\alpha = .05$, we'll call this a rejection, and say there is some meaningful difference in time among treatment groups.

3.)

```
coxph(Surv(T3, D3) ~ Z1, data=burn) -> cox_fit
summary(cox_fit)
```

```
## Call:
## coxph(formula = Surv(T3, D3) ~ Z1, data = burn)
##
##   n= 154, number of events= 48
##
##       coef exp(coef) se(coef)      z Pr(>|z|)
## Z1 -0.5614    0.5704   0.2934 -1.914   0.0557 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##    exp(coef) exp(-coef) lower .95 upper .95
## Z1    0.5704      1.753     0.321     1.014
##
## Concordance= 0.566  (se = 0.039 )
## Likelihood ratio test= 3.73  on 1 df,   p=0.05
## Wald test            = 3.66  on 1 df,   p=0.06
## Score (logrank) test = 3.76  on 1 df,   p=0.05
```

Here, the score test is nearly identical (3.76 on 1 df, as opposed to the K-M's 3.8 on 1), with a similar p-value of .05. The conclusion largely remains the same.

```
cox_full = coxph(
  Surv(T3, D3) ~ Z1 + Z2 + Z3 + Z4 + Z5 + Z6 + Z7 + Z8 + Z9 + Z10  + as.factor(Z11),
  data=burn
)
summary(cox_full)
```

4.)

```
## Call:
## coxph(formula = Surv(T3, D3) ~ Z1 + Z2 + Z3 + Z4 + Z5 + Z6 +
##     Z7 + Z8 + Z9 + Z10 + as.factor(Z11), data = burn)
##
##   n= 154, number of events= 48
##
##                       coef exp(coef)  se(coef)      z Pr(>|z|)
## Z1               -0.651754  0.521131  0.323330 -2.016   0.0438 *
## Z2               -0.556911  0.572976  0.405182 -1.374   0.1693
## Z3                2.149127  8.577367  1.040139  2.066   0.0388 *
## Z4                0.002041  1.002043  0.009843  0.207   0.8357
## Z5               -0.014035  0.986063  0.370920 -0.038   0.9698
## Z6                0.541461  1.718516  0.430265  1.258   0.2082
## Z7               -0.055650  0.945870  0.507956 -0.110   0.9128
## Z8               -0.171817  0.842133  0.393707 -0.436   0.6625
## Z9               -0.324566  0.722841  0.373905 -0.868   0.3854
## Z10               0.228682  1.256943  0.372930  0.613   0.5397
## as.factor(Z11)2   1.527828  4.608156  1.128623  1.354   0.1758
## as.factor(Z11)3   2.192439  8.957029  1.130097  1.940   0.0524 .
## as.factor(Z11)4   0.949734  2.585021  1.036308  0.916   0.3594
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                  exp(coef) exp(-coef) lower .95 upper .95
## Z1                  0.5211     1.9189    0.2765    0.9821
## Z2                  0.5730     1.7453    0.2590    1.2677
## Z3                  8.5774     0.1166    1.1168   65.8752
## Z4                  1.0020     0.9980    0.9829    1.0216
## Z5                  0.9861     1.0141    0.4766    2.0400
## Z6                  1.7185     0.5819    0.7395    3.9939
## Z7                  0.9459     1.0572    0.3495    2.5598
## Z8                  0.8421     1.1875    0.3893    1.8218
## Z9                  0.7228     1.3834    0.3474    1.5042
## Z10                 1.2569     0.7956    0.6052    2.6107
## as.factor(Z11)2     4.6082     0.2170    0.5045   42.0933
## as.factor(Z11)3     8.9570     0.1116    0.9777   82.0549
## as.factor(Z11)4     2.5850     0.3868    0.3391   19.7048
##
## Concordance= 0.739  (se = 0.036 )
## Likelihood ratio test= 27.29  on 13 df,   p=0.01
## Wald test            = 22.39  on 13 df,   p=0.05
## Score (logrank) test = 26.23  on 13 df,   p=0.02
```

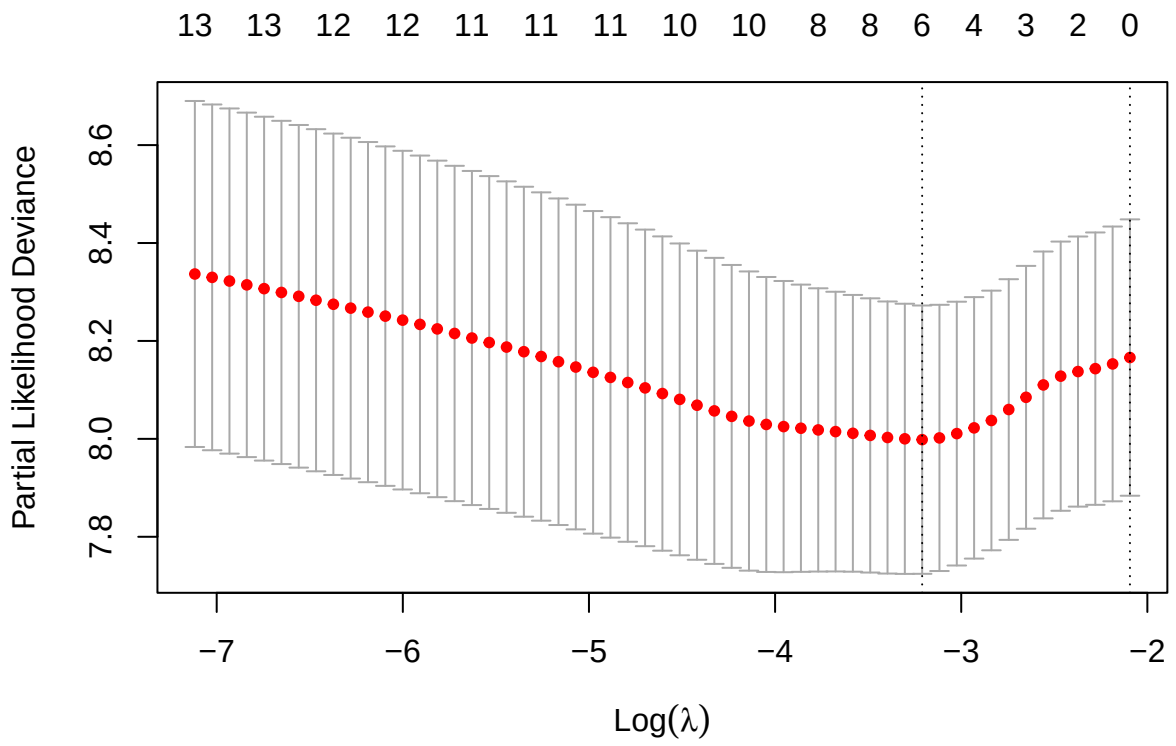
Conditional on all others being included in the model, we see that `Z1`, `Z3`, and to a lesser extent, `Z11` are most helpful. Note that `Z11`, having the four categories `1, 2, 3, 4`, is treated as categorical via one-hot encoding.

5.)

```
M = coxph(
  Surv(T3, D3) ~ Z1 + Z2 + Z3 + Z4 + Z5 + Z6 + Z7 + Z8 + Z9 + Z10  + as.factor(Z11),
  data=burn
```

```
)
D = model.frame(M) # treats diagnosis as continuous
X = model.matrix(M)
Y = D[,1]

cox_lasso = cv.glmnet(X, Y, family='cox')
plot(cox_lasso)
```




```
cox_lasso_lambda_min=glmnet(X, Y, family="cox", lambda=cox_lasso$lambda.min)
cox_lasso_c_idx=cv.glmnet(X, Y, family="cox", type.measure="C")
```

The variables under `lambda.min` are:

```
coef(cox_lasso_lambda_min)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## Z1              -3.665370e-01
## Z2              -2.095169e-01
## Z3               9.177795e-01
## Z4               4.591779e-05
## Z5               .
## Z6               1.044359e-01
## Z7               .
```

```
## Z8                .
## Z9                .
## Z10               .
## as.factor(Z11)2   .
## as.factor(Z11)3   7.612022e-01
## as.factor(Z11)4   .
```

As discussed above, `Z1`, `Z3` and `Z11=3` already looked like strong candidates, given their apparent conditional importance. This suggests that we add `Z2` and `Z6` to that feature pool. Note that the inclusion of only `Z11=3` suggests we should consider binarizing this categorical variable.

6.) Meanwhile, under `C-Index`, we have

```
coef(cox_lasso_c_idx)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## Z1               -0.17526277
## Z2               -0.03600325
## Z3                0.58365684
## Z4                .
## Z5                .
## Z6                .
## Z7                .
## Z8                .
## Z9                .
## Z10               .
## as.factor(Z11)2   .
## as.factor(Z11)3   0.51156426
## as.factor(Z11)4   .
```

Here, we only select `Z1`, `Z3` and `Z11=3`.

7.) If we want to take an all-or-none approach to the three `Z11` indicators (which combined to make a one-hot-encoded `Z11`), we might turn to group lasso, and minimize w.r.t.

$$\ell(X\beta) + \lambda_1 ||\beta_{Z \neq 11}||_1 + \lambda_2 ||\beta_{Z=11}||_2^{1/2}$$

Bone Marrow

1.)

Look at the data:

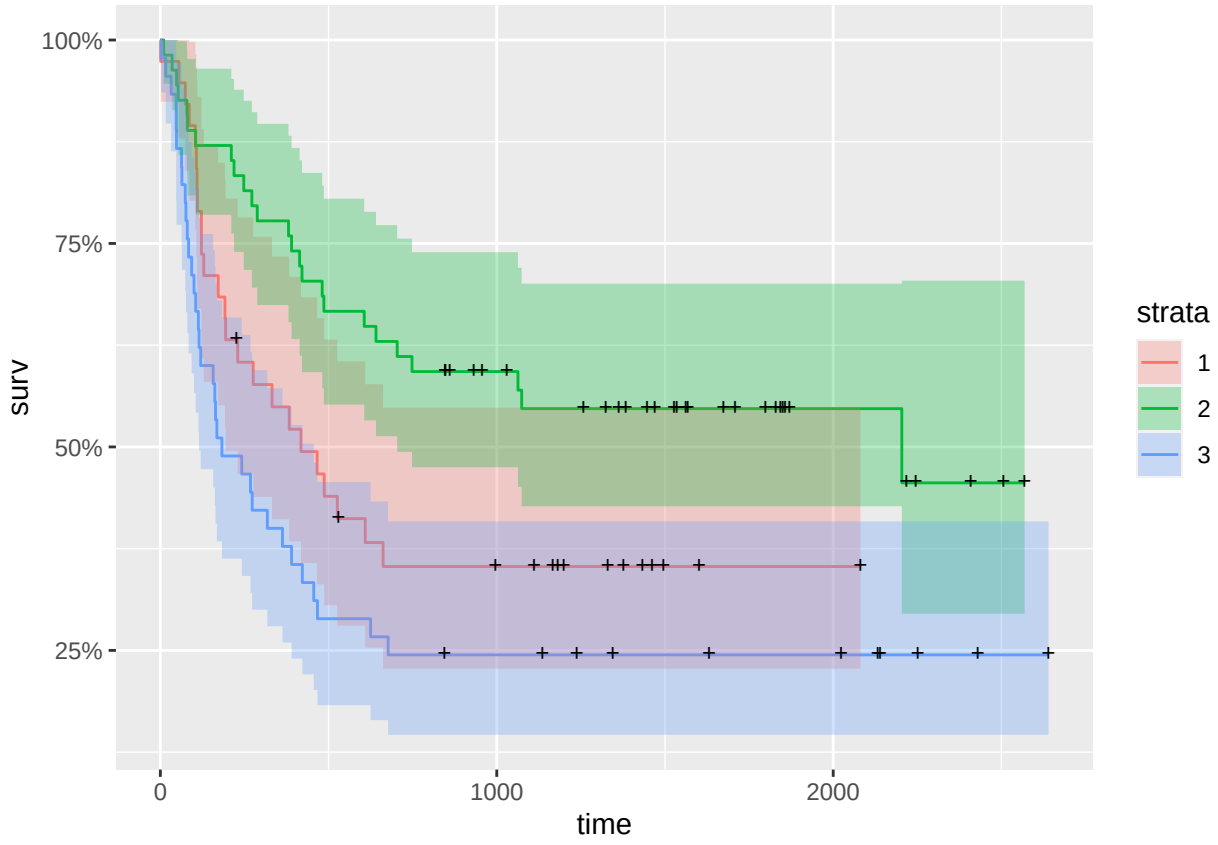
```
data(bmt)
head(bmt)
```

```
##   group   t1   t2 d1 d2 d3   ta da  tc dc tp dp z1 z2 z3 z4 z5 z6   z7 z8 z9
## 1     1 2081 2081  0  0  0   67  1 121  1 13  1 26 33  1  0  1  1   98  0  1
## 2     1 1602 1602  0  0  0 1602  0 139  1 18  1 21 37  1  1  0  0 1720  0  1
## 3     1 1496 1496  0  0  0 1496  0 307  1 12  1 26 35  1  1  1  0  127  0  1
## 4     1 1462 1462  0  0  0   70  1  95  1 13  1 17 21  0  1  0  0  168  0  1
## 5     1 1433 1433  0  0  0 1433  0 236  1 12  1 32 36  1  1  1  1   93  0  1
```

```
## 6     1 1377 1377   0   0   0 1377   0 123   1 12   1 22 31   1   1   1   1 2187   0   1
##    z10
## 1    0
## 2    0
## 3    0
## 4    0
## 5    0
## 6    0
```

Now fit the K-M curves:

```
survfit(Surv(t2, d3) ~ group, data=bmt) -> km_fit
autoplot(km_fit)
```



2.)

Test results are shown at the bottom of the summary:

```
survdiff(Surv(t2, d3) ~ group, data=bmt)
```

```
## Call:
## survdiff(formula = Surv(t2, d3) ~ group, data = bmt)
##
##          N Observed Expected (O-E)^2/E (O-E)^2/V
```

```
## group=1 38       24     21.9     0.211     0.289
## group=2 54       25     40.0     5.604    11.012
## group=3 45       34     21.2     7.756    10.529
##
##  Chisq= 13.8  on 2 degrees of freedom, p= 0.001
```

A p-value of .001 indicates a clear rejection, and there is likely some difference in survival among groups.

As set forth in the Survival I slides, we have, for any time point j:

- rows (the two dimensional – using the HW framing here, as the slides transpose the table to 3x2) correspond to (i) counts of deaths up to time T_j and (2) counts of survivals past time T_j. In other words, rows loosely represent a "check-in" at time j, where we tally up deaths and survivals at that point.
- columns (the three dimensional) correspond to `group`.
- We do this over every sampled time point j, so K equals the number of unique timestamps (across all groups) we collected.

Whereas the hypergeometric distribution falls nicely out of the 2x2 case, here we generalize to a multivariate hypergeometric (as it's a 3x2, or perhaps larger table), wherein the number of deaths/non-survivals within a group (i.e. failures) corresponds to a three-binned urn. That is, instead of counting ways we could put marbles into one bin or the other, we're divvying up how they go into three bins (see: https://en.wikipedia.org/wiki/Hypergeometric_distribution#Multivariate_hypergeometric_distribution)

3.)

As detailed in the `survival::survdiff` documentation, the `exp` attribute provides the expected number of events (deaths) within each group. That is, you fit the model on each observation. Then, for each time T_j, you compute the expected number of deaths up to that point via (for group k)

$$\delta_{k_j,j} \frac{|R_k(T_j)}{|R(T_j)|}$$

You then take the last of these (i.e. how long your survivors were allowed to survive, the survival opportunity window) – this is your expected number of deaths in the time period for the group. You can compare that to the expected.

In a similar spirit, the `var` attribute just gives the variance between the aforementioned counts – for example, with three groups, there'll be three expected counts, and hence a 3×3 covariance. The diagonal of these are variances.

Now, for your survival opportunity window, you have observed counts, expected counts (above), and variances of these expected counts (from the covariance). This lays the groundwork for the chi-squared test, i.e. for each group k computing $(O_k - E_k/V_k)^2$, summing it up, and using that as your test statistic. The test statistic itself was derived in class.

4.)

```
empty_fit = coxph(Surv(ta, da) ~ 1, data=bmt)
full_fit = coxph(Surv(ta, da) ~ z10 * as.factor(group), data=bmt)
step_fit = step(empty_fit, scope=formula(full_fit), direction="forward", trace=2)
```

```
## Start:  AIC=248.7
## Surv(ta, da) ~ 1
##
## trying +z10
## trying +as.factor(group)
##                     Df    AIC
## <none>                 248.70
## + z10                1 250.26
## + as.factor(group)   2 251.35
```

```
# step_fit = stepAIC(empty_fit, scope=formula(full_fit), direction="forward", trace=2)
step_fit$anova
```

```
##   Step Df Deviance Resid. Df Resid. Dev      AIC
## 1      NA       NA        26   248.6971 248.6971
```

Unfortunately, the interaction is not selected – in fact no model is selected. However, when we begin a stepwise regression with `~ z10 * group`, and consider the other covariates (`z1 - z10`), we get (allowing all the z's to interact with group)

```
actually_full_fit = coxph(Surv(ta, da) ~ (
  z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9 + z10) * as.factor(group),
  data=bmt
)
step_fit_2 = step(full_fit, scope=formula(actually_full_fit), direction="forward", trace=2)
```

```
## Start:  AIC=255.7
## Surv(ta, da) ~ z10 * as.factor(group)
##
## trying +z1
## trying +z2
## trying +z3
## trying +z4
## trying +z5
## trying +z6
## trying +z7
## trying +z8
## trying +z9
##        Df    AIC
## + z1    1 249.06
## + z2    1 250.59
## + z6    1 253.79
## <none>    255.70
## + z9    1 256.84
## + z4    1 256.99
## + z3    1 257.45
## + z5    1 257.45
## + z8    1 257.55
## + z7    1 257.68
##
## Step:  AIC=249.06
## Surv(ta, da) ~ z10 + as.factor(group) + z1 + z10:as.factor(group)
```

```
##
## trying +z2
## trying +z3
## trying +z4
## trying +z5
## trying +z6
## trying +z7
## trying +z8
## trying +z9
## trying +z1:as.factor(group)
##                         Df    AIC
## + z6                     1 248.54
## <none>                     249.06
## + z4                     1 250.38
## + z2                     1 250.45
## + z9                     1 250.57
## + z7                     1 250.72
## + z8                     1 250.74
## + z3                     1 250.89
## + z5                     1 251.06
## + z1:as.factor(group)    2 251.66
##
## Step:  AIC=248.54
## Surv(ta, da) ~ z10 + as.factor(group) + z1 + z6 + z10:as.factor(group)
##
## trying +z2
## trying +z3
## trying +z4
## trying +z5
## trying +z7
## trying +z8
## trying +z9
## trying +z1:as.factor(group)
## trying +z6:as.factor(group)
##                         Df    AIC
## <none>                     248.54
## + z4                     1 249.78
## + z8                     1 249.80
## + z2                     1 249.84
## + z1:as.factor(group)    2 249.87
## + z9                     1 250.17
## + z7                     1 250.25
## + z5                     1 250.28
## + z3                     1 250.47
## + z6:as.factor(group)    2 251.45
```

```
step_fit_2$anova
```

```
##   Step Df Deviance Resid. Df Resid. Dev      AIC
## 1      NA       NA        21   245.6968 255.6968
## 2 + z1 -1 8.632976        20   237.0638 249.0638
## 3 + z6 -1 2.527056        19   234.5367 248.5367
```

```
final_fit = coxph(formula(step_fit_2), data=bmt)
```

Here, the stepwise regression adds in `z1` and `z6` (consistent with their important utility noted above).

```
summary(final_fit)
```

```
## Call:
## coxph(formula = formula(step_fit_2), data = bmt)
##
##   n= 137, number of events= 26
##
##                           coef exp(coef) se(coef)      z Pr(>|z|)
## z10                    -0.55326   0.57507  0.70815 -0.781  0.43464
## as.factor(group)2      -0.36099   0.69698  0.54594 -0.661  0.50846
## as.factor(group)3      -1.15320   0.31563  0.65937 -1.749  0.08030 .
## z1                      0.06393   1.06601  0.02311  2.766  0.00568 **
## z6                      0.64861   1.91289  0.41065  1.579  0.11423
## z10:as.factor(group)2 -0.69359   0.49978  1.26930 -0.546  0.58476
## z10:as.factor(group)3  0.18590   1.20430  1.14454  0.162  0.87097
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                       exp(coef) exp(-coef) lower .95 upper .95
## z10                      0.5751     1.7389   0.14353     2.304
## as.factor(group)2        0.6970     1.4348   0.23907     2.032
## as.factor(group)3        0.3156     3.1683   0.08668     1.149
## z1                       1.0660     0.9381   1.01880     1.115
## z6                       1.9129     0.5228   0.85535     4.278
## z10:as.factor(group)2    0.4998     2.0009   0.04153     6.015
## z10:as.factor(group)3    1.2043     0.8304   0.12779    11.349
##
## Concordance= 0.683  (se = 0.055 )
## Likelihood ratio test= 14.16  on 7 df,   p=0.05
## Wald test            = 13.7  on 7 df,   p=0.06
## Score (logrank) test = 13.82  on 7 df,   p=0.05
```

As shown above, the main effect for `MTX=z10` on its own takes coefficient -0.55326. The CI for this coefficient is

```
-.5536 - 1.96 * 0.70815; -.5536 + 1.96 * 0.70815
```

```
## [1] -1.941574
```

```
## [1] 0.834374
```

Case 1: `group = 1`.

Exponentiated, this amounts to .5751 with CI $(.14353, .17389)$. In other words, this means pretty squarely that when `z10` is 1, the hazard decreases for `group = 1`, the baseline group, which is *good* for the survival of the patient.

Case 2: `group = 2`.

Then, when `group = 2`, we have linear contribution $\beta_{z10} + \beta_{z10,G2} = 0.69359 - 0.55326 = -1.24685$. As $\beta_{z10,G2} < 0$, this makes $\beta_{z10} + \beta_{z10,G2} < \beta_{z10}$, and makes the hazard even smaller. So with `group=2`, the protection is even stronger (CI omitted since covariance of coefficients not provided, which is necessary for the CI).

Case 3: `group = 3`.

Lastly, when `group = 3`, we have linear contribution $\beta_{z10} + \beta_{z10,G3} = 1.2043 - 0.55326 = .65104$. This increases the linear contribution, in turn increasing the hazard, and predicting that the patient is less likely to survive (as their hazard has increase) as compared to a group 1 or 2 patient.

Of course, there's little confidence to be had here. All of the confidence intervals, upon exponentiation, contain 1, so we can't say with much force that hazard is increasing or decreasing. A null of no change in hazard would fail to be rejected.

Conditional Independence

a.)

Leaning conditional independence here, as features may have relationship with data availability, and in turn a relationship with censoring? Consider the following (bad) example: let's say you're doing a survival study of football players, and the survival event equals first concussion and time is measured in snaps taken. Here, weight might reasonably be a feature in your model – if you're bigger, maybe you're more likely to be the hammer and not the nail. However, at the same time, players at lower weights may see the field less. As a result, these players might play fewer snaps, and by the end of the season, they'd hardly have had a chance to get concussed. In this way, the feature `X` would be related to `D` – suggesting that conditional independence may be well-suited in certain cases.

b.)

Consider the $\delta = 1$ case first. We have, using M, m instead of O, o for notational convenience

$$\begin{aligned} P_\beta(M \in [m, m+dm], \delta = 1|X) &= P_\beta(T \in [m, m+dm], C > T|X) \\ &= \int_{v=0}^{\infty} \int_{u=0}^{\infty} f^{(\beta)}_{C,T|X}(u, v)\delta(v \in [m, m+dm])\delta(u > v)duduv \\ &= \int_{v=m}^{m+dm} \int_{u=v}^{\infty} f^{(\beta)}_{C,T|X}(u, v)dudv \end{aligned}$$

Now, if we have conditional independence, we may separate $f_{C,T|X}$ and continue with the proof. Otherwise, as is the case for independence (which does not imply cond. independence), we are stuck here, unless $f_{C,T|X}$ is an especially nice density (unlikely) to integrate over. Now, proceeding forward *with* conditional independence, we can then do

$$
\begin{aligned}
P_\beta(M \in [m, m+dm], \delta = 1|X) &= P_\beta(T \in [m, m+dm], C > T|X) \\
&= \int_{v=0}^{\infty} \int_{u=0}^{\infty} f^{(\beta)}_{C,T|X}(u,v)\delta(v \in [m, m+dm])\delta(u > v)dudv \\
&= \int_{v=m}^{m+dm} \int_{u=v}^{\infty} f^{(\beta)}_{C,T|X}(u,v)dudv \\
&= \int_{v=m}^{m+dm} \int_{u=v}^{\infty} f_{C|X}(u) f^{(\beta)}_{T|X}(v)dudv \\
&= \int_{v=m}^{m+dm} f^{(\beta)}_{T|X}(v) \int_{u=v}^{\infty} f_{C|X}(u)dudv \\
&= \int_{v=m}^{m+dm} f^{(\beta)}_{T|X}(v) \int_{u=v}^{\infty} f_{C|X}(u)dudv \\
&= \int_{v=m}^{m+dm} f^{(\beta)}_{T|X}(v)(1 - F_{C|X}(v))dv.
\end{aligned}
$$

As noted in OH, if we then assume dm gets small and that (think Riemann)

$$\int_{v=m}^{m+dm} (1 - F_{C|X}(v))dv \approx (1 - F_{C|X}(m))dm$$

we may proceed with

$$\int_{v=m}^{m+dm} f^{(\beta)}_{T|X}(v)(1 - F_{C|X}(v))dv \approx (1 - F_{C|X}(v))dm \cdot \int_{v=m}^{m+dm} f^{(\beta)}_{T|X}(v) \approx (1 - F_{C|X}(v))dm \cdot h_\beta(m),$$

based on how the hazard h was originally defined. This is much more tractable, and the approximation follows from lecture slides `Survival-Analysis-4`. The same observation holds in reverse for the $\delta = 0$ case, i.e.

$$
\begin{aligned}
P_\beta(M \in [m, m+dm], \delta = 0|X) &= P_\beta(C \in [m, m+dm], C < T|X) \\
&= \int_{u=0}^{\infty} \int_{v=0}^{\infty} f^{(\beta)}_{C,T|X}(u,v)\delta(u \in [m, m+dm])\delta(u < v)dvdu \\
&= \int_{u=m}^{m+dm} \int_{v=u}^{\infty} f^{(\beta)}_{C,T|X}(u,v)dvdu \\
&= \int_{u=m}^{m+dm} \int_{v=u}^{\infty} f_{C|X}(u) f^{(\beta)}_{T|X}(v)dvdu \\
&= \int_{u=m}^{m+dm} f_{C|X}(u) \int_{v=u}^{\infty} f^{(\beta)}_{T|X}(v)dvdu \\
&= \int_{u=m}^{m+dm} f_{C|X}(u)(1 - F^{(\beta)}_{T|X}(u))du \\
&= \int_{u=m}^{m+dm} f_{C|X}(u)S_\beta(u)du.
\end{aligned}
$$

As before, if we *do not have conditional independence, we are stuck at the third line of the proof. We need conditional independence to go through!* As before, if we say that as dm gets small, we can go

$$\int_{u=m}^{m+dm} S_\beta(u)du \approx S_\beta(m)dm$$

and hence

$$\int_{v=m}^{m+dm} f_{C|X}(u)S_\beta(u)du \approx S_\beta(m)dm \int_{v=m}^{m+dm} f_{C|X}(u)du \approx S_\beta(m)dm \cdot P_{C|X}(C \in [m, m+dm])$$

and we can interpret $\int_{v=m}^{m+dm} f_{C|X}(u)du$ to be a probability if we need interpretability.

From class, our likelihood is

$$\begin{aligned} L(\beta, X_i, M_i) = \prod_{i=1}^{n} h_\beta(X_i, M_i)^{\delta_i} S_\beta(X_i, M_i) & \\ &= \prod_{i=1}^{n} h_\beta(X_i, M_i)^{\delta_i} S_\beta(X_i, M_i)^{\delta_i} S_\beta(X_i, M_i)^{1-\delta_i} \\ &= \prod_{i=1}^{n} f_\beta(X_i, M_i)^{\delta_i} S_\beta(X_i, M_i)^{1-\delta_i}, \end{aligned}$$

which for each term in the product, are effectively are our two cases as presented (albeit in integral-chunk form) above: the $f_\beta(X_i, M_i)^{\delta_i}$ term corresponds to the $\delta = 1$ case at the start of the problem, while the $S_\beta(X_i, M_i)^{1-\delta_i}$ corresponds to the $\delta = 0$ case examined shortly thereafter.

So in closing – we showed above that when we have conditional independence, we can recover the right-censored likelihood we've been using. Of course, for all of this to work, we have to hold onto a few more assumptions, to wit:

- Censored patients would have had a similar survival profile as those who we got to see die, at all times in the data.
- Proportioanl hazards
- No confounding over the study, i.e. there was no medicine introduced (or something equivalent) that later subjects more robust to survival.