Final Report

due November 16, 2021 by 11:59 PM

Enzo, Layla, Madeleine

November 16, 2021

```r
cities <- initial_data %>%
  select(StateAbbr, PlaceName, Population2010, PlaceFIPS, Geolocation,
         ACCESS2_AdjPrev,CANCER_AdjPrev, CHD_AdjPrev, CHECKUP_AdjPrev,
         COPD_AdjPrev, COLON_SCREEN_AdjPrev,COREM_AdjPrev, COREW_AdjPrev,
         KIDNEY_AdjPrev, MAMMOUSE_AdjPrev, PAPTEST_AdjPrev) %>%
  dplyr::rename(state = StateAbbr, city = PlaceName, population = Population2010,
         health_access = ACCESS2_AdjPrev, cancer = CANCER_AdjPrev,
         heart_disease = CHD_AdjPrev, checkup = CHECKUP_AdjPrev,
         chronic_lung_disease = COPD_AdjPrev,
         colon_screen = COLON_SCREEN_AdjPrev,
         men_colorectal_cancer_screen = COREM_AdjPrev,
         women_colorectal_cancer_screen = COREW_AdjPrev,
         chronic_kidney_disease = KIDNEY_AdjPrev,
         mammogram = MAMMOUSE_AdjPrev,
         pap_test = PAPTEST_AdjPrev) %>%
  mutate(large_metro = (population >= 1500000),
         metro = (population >= 500000 & population < 1500000),
         med_urban = (population >= 200000 & population < 500000),
         small_urban = (population >= 50000 & population < 200000)) %>%
  mutate(west = (state %in% c("WA", "OR", "ID", "MT", "WY", "CA", "NV", "UT",
                              "CO", "AZ", "NM")),
         midwest = (state %in% c("ND", "SD", "NE", "KS", "MN", "IA", "MO",
                                 "WI", "IL", "IN", "MI", "OH")),
         northeast = (state %in% c("PA", "NY", "VT", "NH", "MA", "CT", "RI",
                                   "NJ", "ME", "DC")),
         south = (state %in% c("OK", "TX", "AR", "LA", "MS", "AL", "TN", "KY",
                               "WV", "VA", "MD", "DE", "NC", "SC", "GA",
                               "FL"))) %>%
  mutate(region = ifelse(west == TRUE, "West",
                         ifelse(midwest == TRUE, "Midwest",
                                ifelse(northeast == TRUE, "Northeast",
                                       ifelse(south == TRUE,
                                              "South", NA))))) %>%
  mutate(city_size = ifelse(large_metro == TRUE, "Large Metropolitan",
                         ifelse(metro == TRUE, "Metropolitan",
                                ifelse(med_urban == TRUE, "Medium-Size Urban",
                                       ifelse(small_urban == TRUE,
                                              "Small-Size Urban", NA))))) %>%
  na.omit(city_size) %>%
  na.omit(region) %>%
  mutate(checkup_n = (population*(checkup/100)),
         colon_n_screened = (population*(colon_screen/100)),
         m_colorectal_n_screened = (population*(men_colorectal_cancer_screen/100)),
         w_colorectal_n_screened =
           (population*(women_colorectal_cancer_screen/100)),
         mammogram_n_screened = (population*(mammogram/100)),
         pap_n_screened = (population*(pap_test/100))) %>%
  mutate(no_checkup_n = (population - checkup_n),
         no_colon_n_screened = (population - colon_n_screened),
         no_m_colorectal_n_screened = (population - m_colorectal_n_screened),
         no_w_colorectal_n_screened =
           (population - w_colorectal_n_screened),
         no_mammogram_n_screened = (population - mammogram_n_screened),
         no_pap_n_screened = (population - pap_n_screened))

# checked the residual plot withouth this and need todo this to make model more accurate
#this is the training data set called model_cities
set.seed(100)

split <- initial_split(cities, prop = 3/4, strata = region)

model_cities <- training(split)
cities_test <- testing(split)

#for introductory visualizations
west_cities <- cities %>%
  filter(west)

midwest_cities <- cities %>%
  filter(midwest)

northeast_cities <- cities %>%
  filter(northeast)

south_cities <- cities %>%
  filter(south)
```

#boxplot of region and city size

Introductory Visualisations of Data Set

```
#showing strong correlation between checkup and the other three outcome variables - paptest, colon scre
correlation1 <- cor.test(cities$checkup_n, cities$pap_n_screened,
                         method = "pearson")
correlation2 <- cor.test(cities$checkup_n, cities$colon_n_screened,
                         method = "pearson")
correlation3 <- cor.test(cities$checkup_n, cities$mammogram_n_screened,
                         method = "pearson")
print(correlation1)
```

```
##
##  Pearson's product-moment correlation
##
## data:  cities$checkup_n and cities$pap_n_screened
## t = 332.26, df = 449, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9975606 0.9983151
## sample estimates:
##       cor
## 0.9979726
```

```
print(correlation2)
```

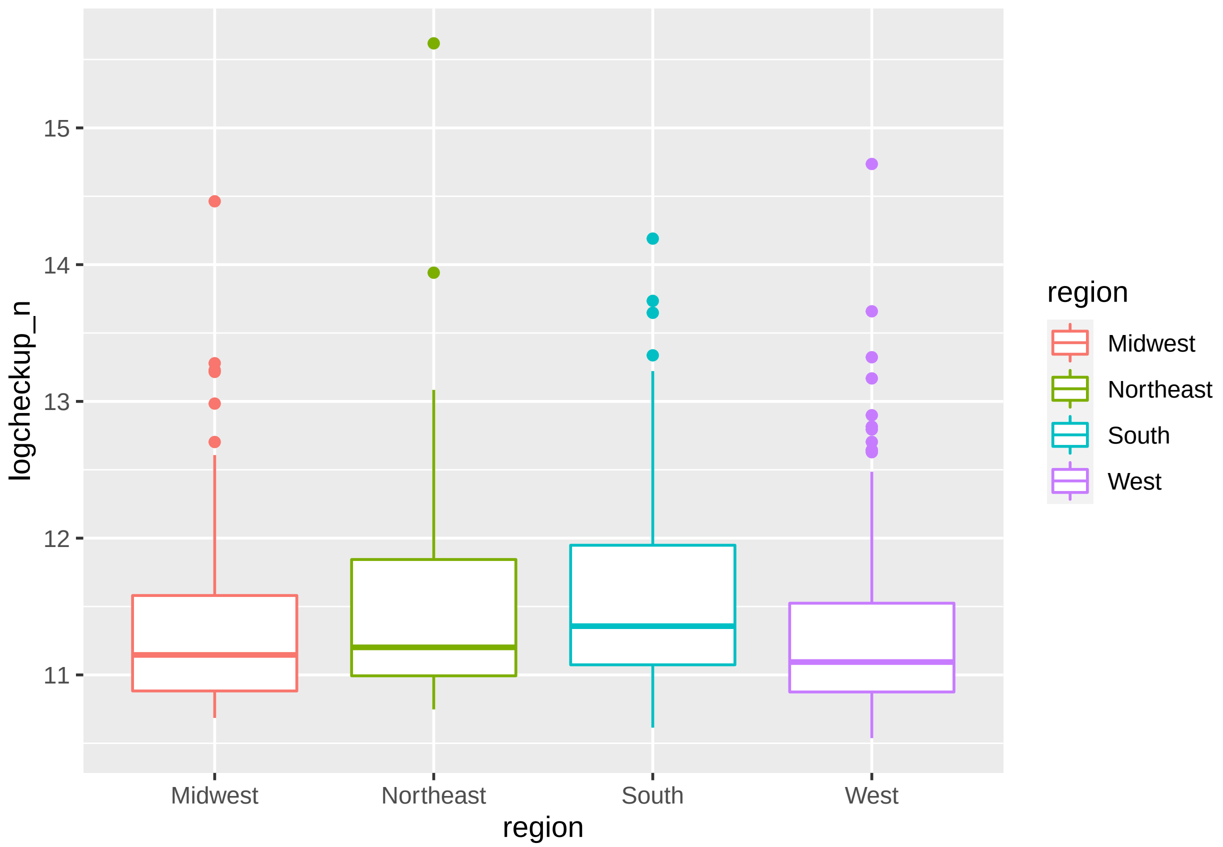
```
##
##  Pearson's product-moment correlation
##
## data:  cities$checkup_n and cities$colon_n_screened
## t = 294.91, df = 449, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9969063 0.9978629
## sample estimates:
##       cor
## 0.9974287
```

```
print(correlation3)
```

```
##
##  Pearson's product-moment correlation
##
## data:  cities$checkup_n and cities$mammogram_n_screened
## t = 437.63, df = 449, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9985920 0.9990276
## sample estimates:
##       cor
## 0.9988299
```

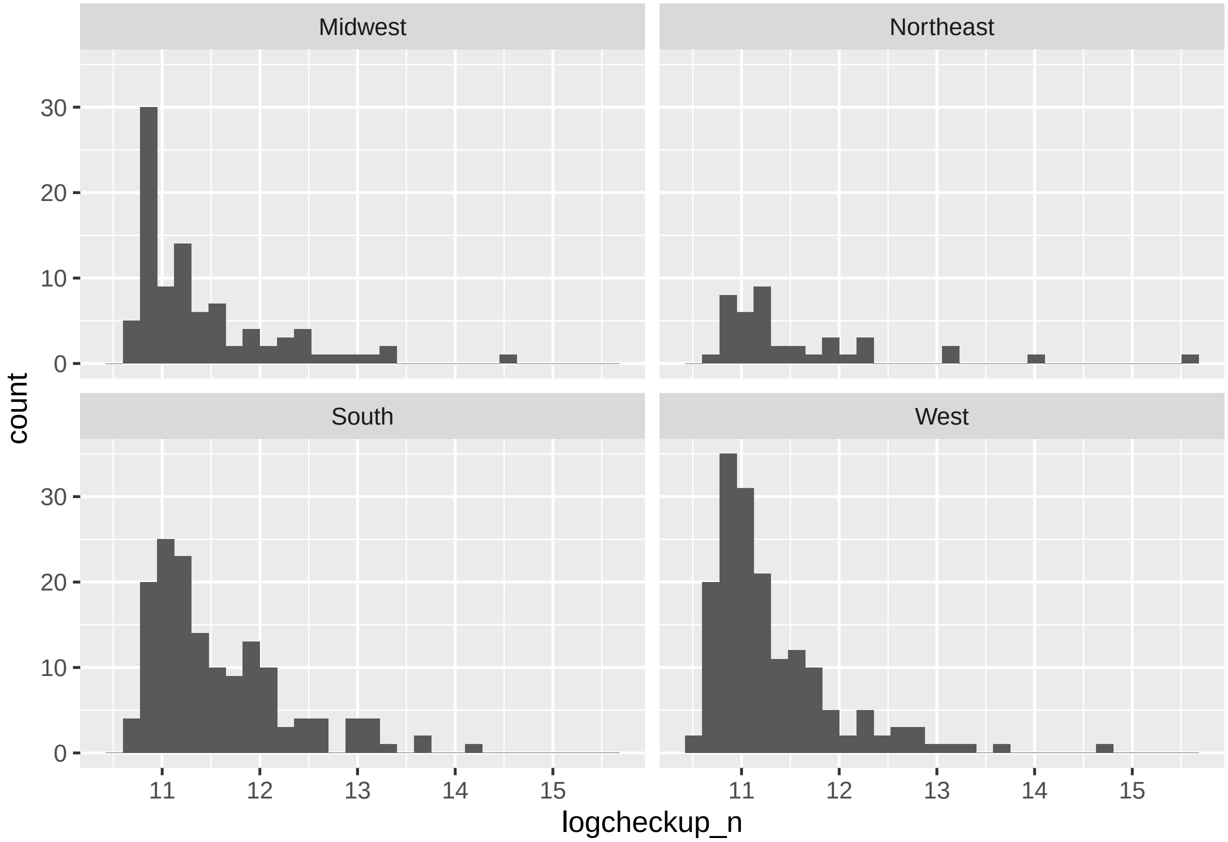
```
#wrangling for visualizations
cities2 <- cities %>%
  filter( region != "NA") %>%
  mutate(logcheckup_n = log(checkup_n))

#(1) boxplots of checkup (n and log) by region
ggplot(data = cities2,
       aes(x = region, y = logcheckup_n,
           color = region)) +
  geom_boxplot()
```



```
#(2) histograms showing mean checkup (not that well), faceted by region
cities2 %>%
ggplot(aes(x = logcheckup_n)) +
  facet_wrap(~region) +
  geom_histogram()
```

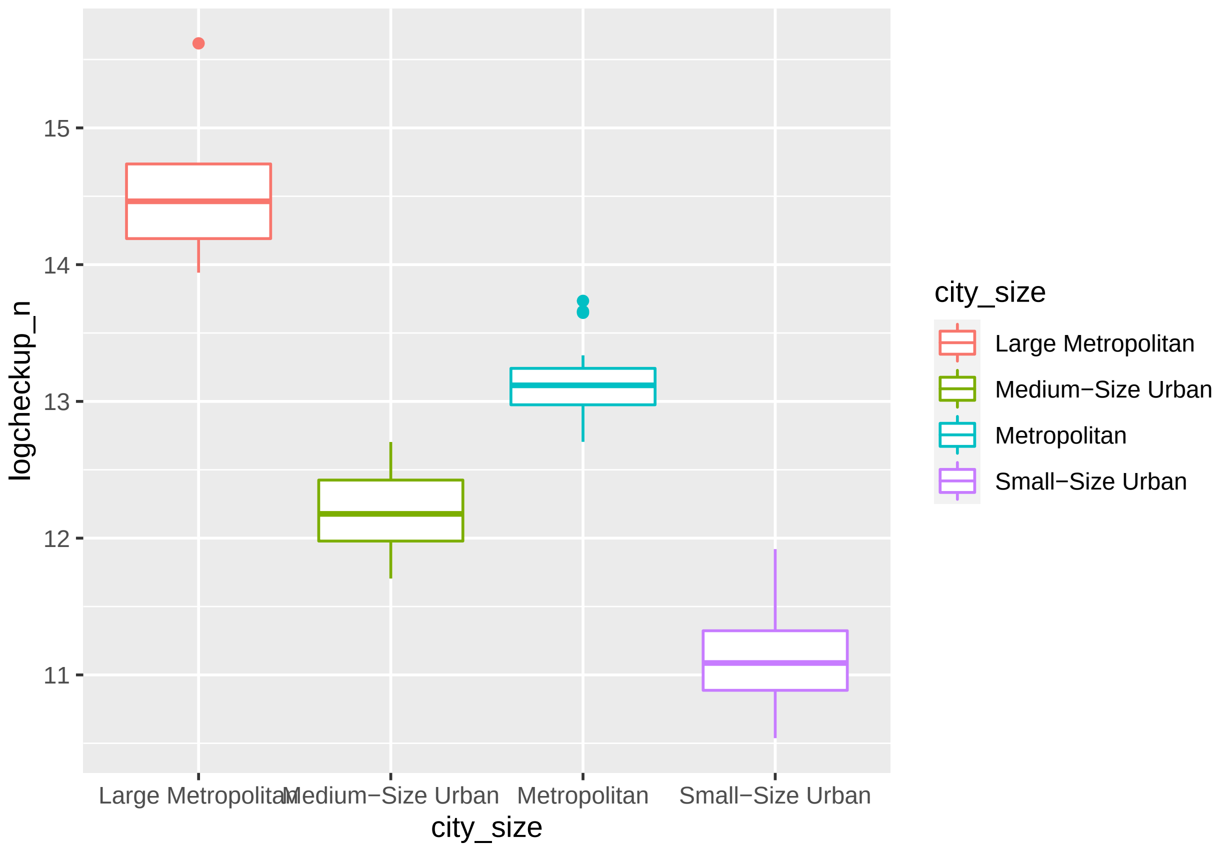
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
#(3) anova shows means of checkup (n and log) are different among the regions
anova1 <- aov(logcheckup_n ~ region, data = cities2)
tidy(anova1)
```

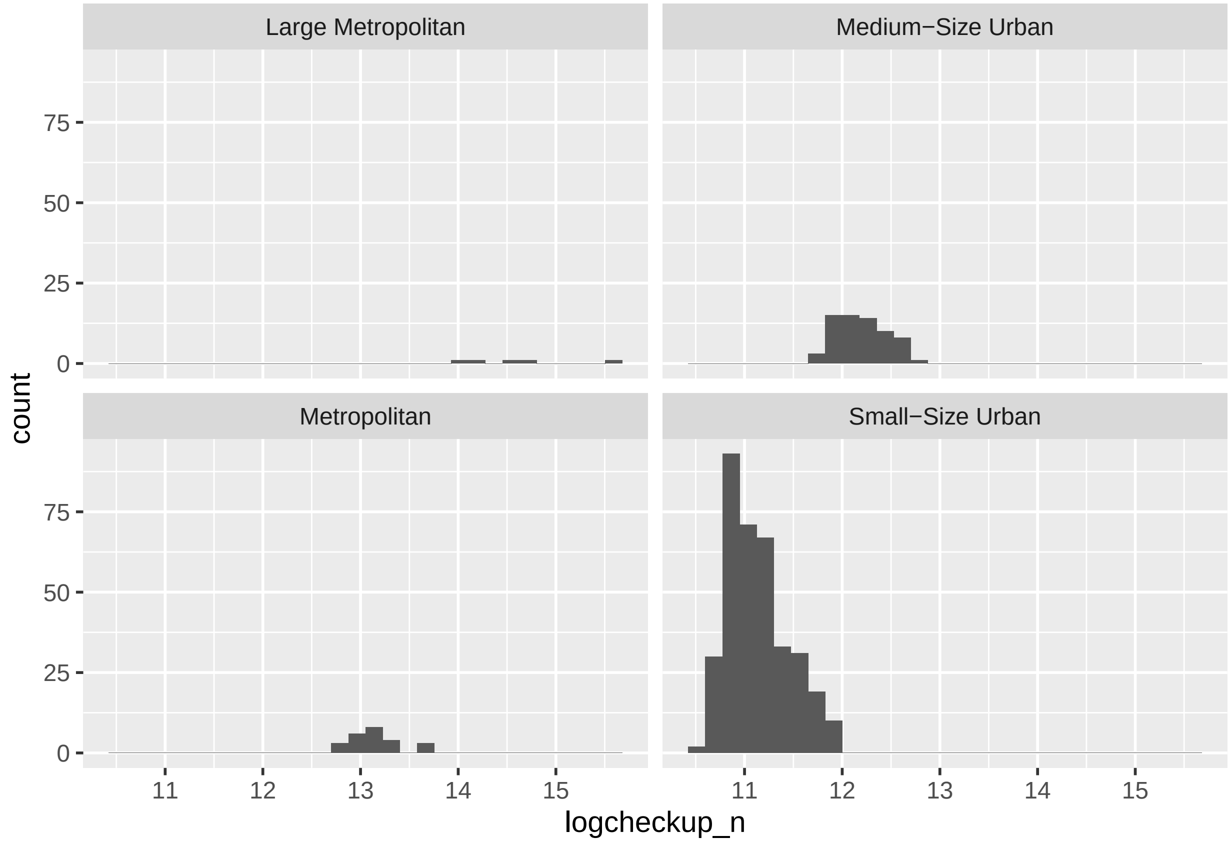
```
## # A tibble: 2 x 6
##   term          df  sumsq meansq statistic  p.value
##   <chr>      <dbl>  <dbl>  <dbl>     <dbl>    <dbl>
## 1 region         3   6.32  2.11       4.16  0.00638
## 2 Residuals    447 226.    0.507     NA     NA
```

```
#(4) boxplots of checkup (n and log) by city size
ggplot(data = cities2,
       aes(x = city_size, y = logcheckup_n,
           color = city_size)) +
  geom_boxplot()
```



```
#(5) histograms showing mean checkup, faceted by city size
cities2 %>%
ggplot(aes(x = logcheckup_n)) +
  facet_wrap(~city_size) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



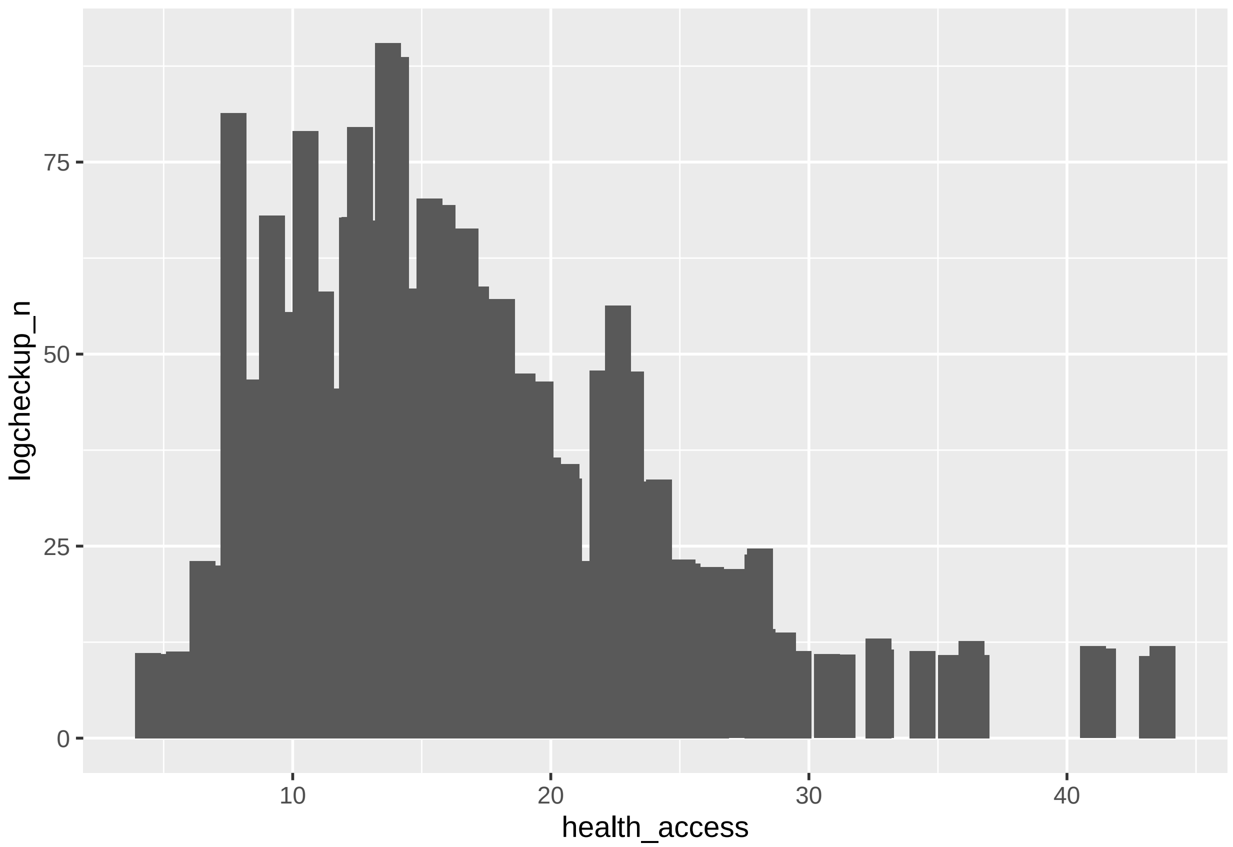
```
#(6) anova shows means of checkup (n and log) are different among the city sizes
anova2 <- aov(logcheckup_n ~ city_size, data = cities2)
tidy(anova2)
```

```
## # A tibble: 2 x 6
##   term         df sumsq  meansq statistic    p.value
##   <chr>     <dbl> <dbl>   <dbl>     <dbl>      <dbl>
## 1 city_size     3 191.   63.5        671.  4.15e-165
## 2 Residuals   447  42.3   0.0946      NA   NA
```

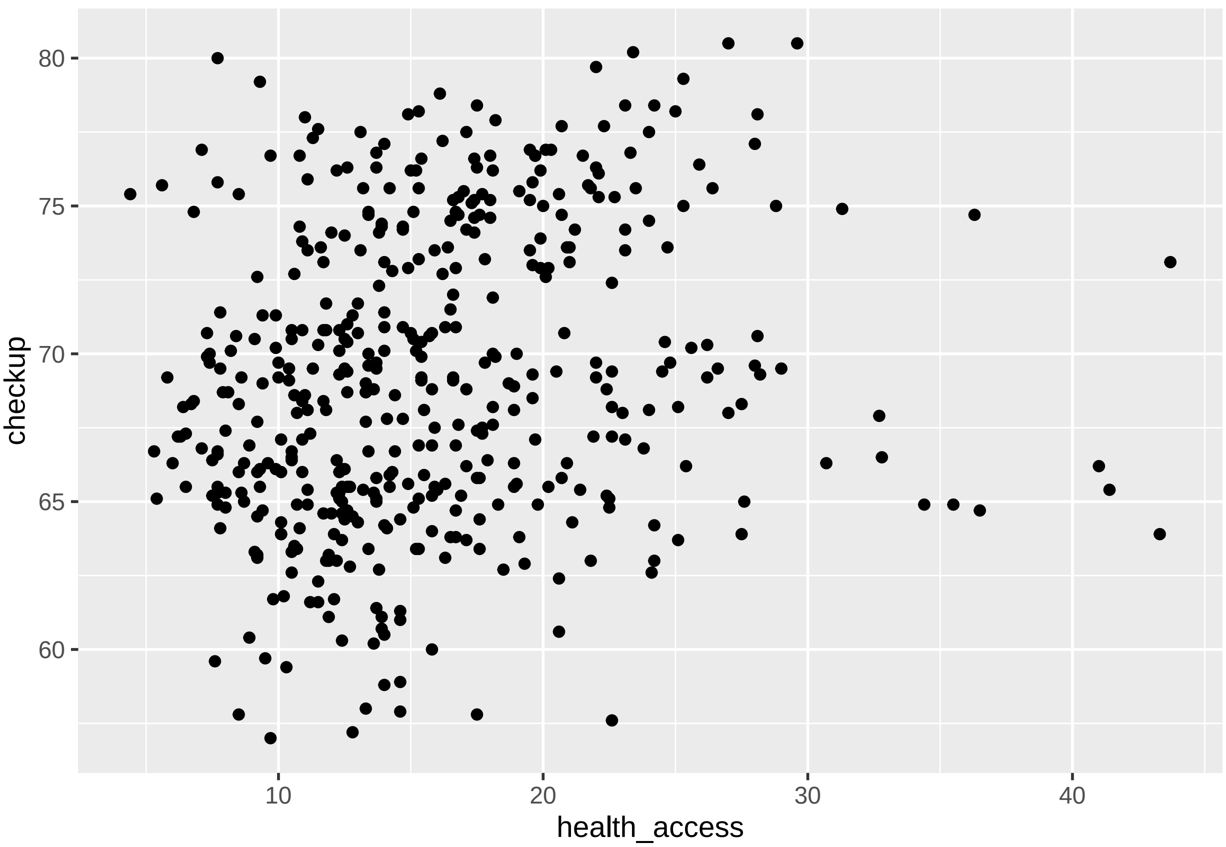
```
#(7) enzo's graph for health access vs checkup
ggplot(data = cities2,
       aes(x = health_access, y = logcheckup_n)) +
  geom_col(aes(width = 1))
```

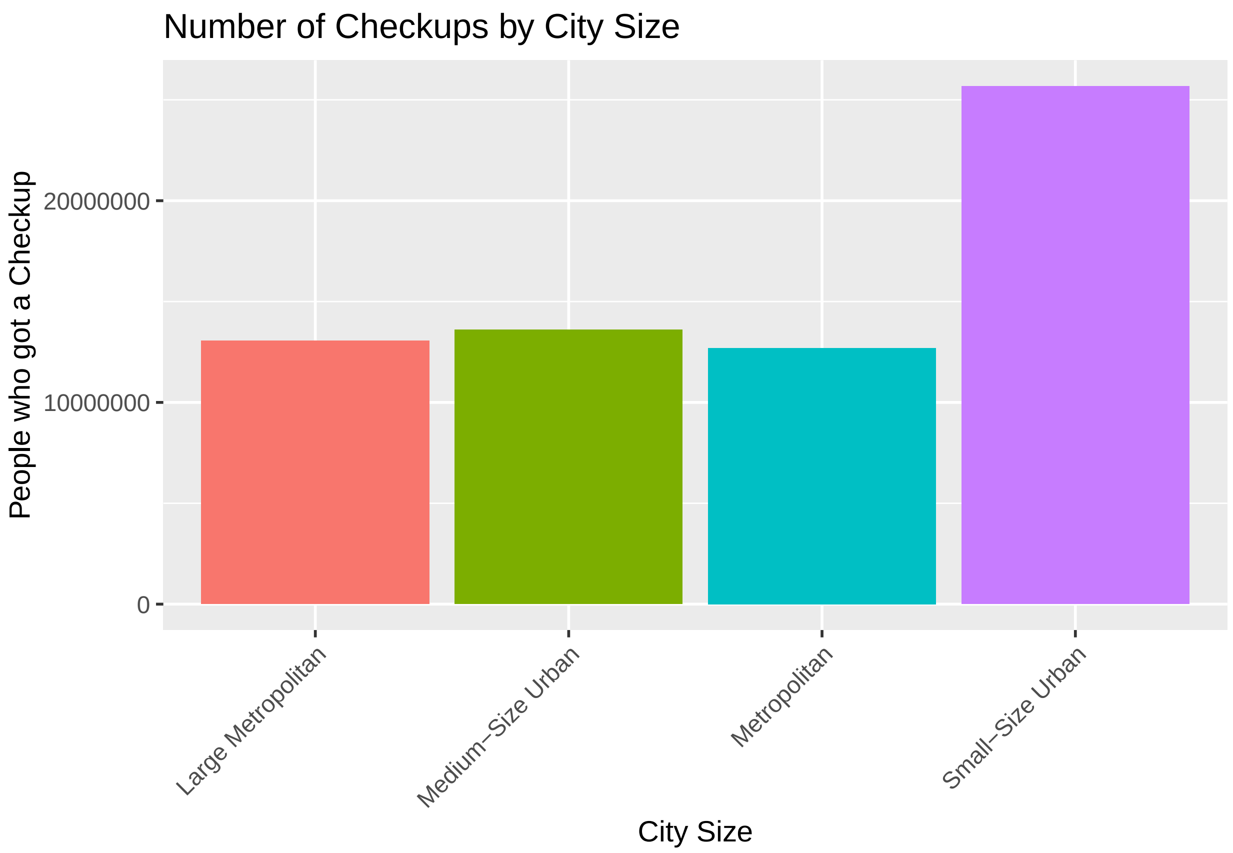
```
## Warning: Ignoring unknown aesthetics: width
```

```
## Warning: position_stack requires non-overlapping x intervals
```



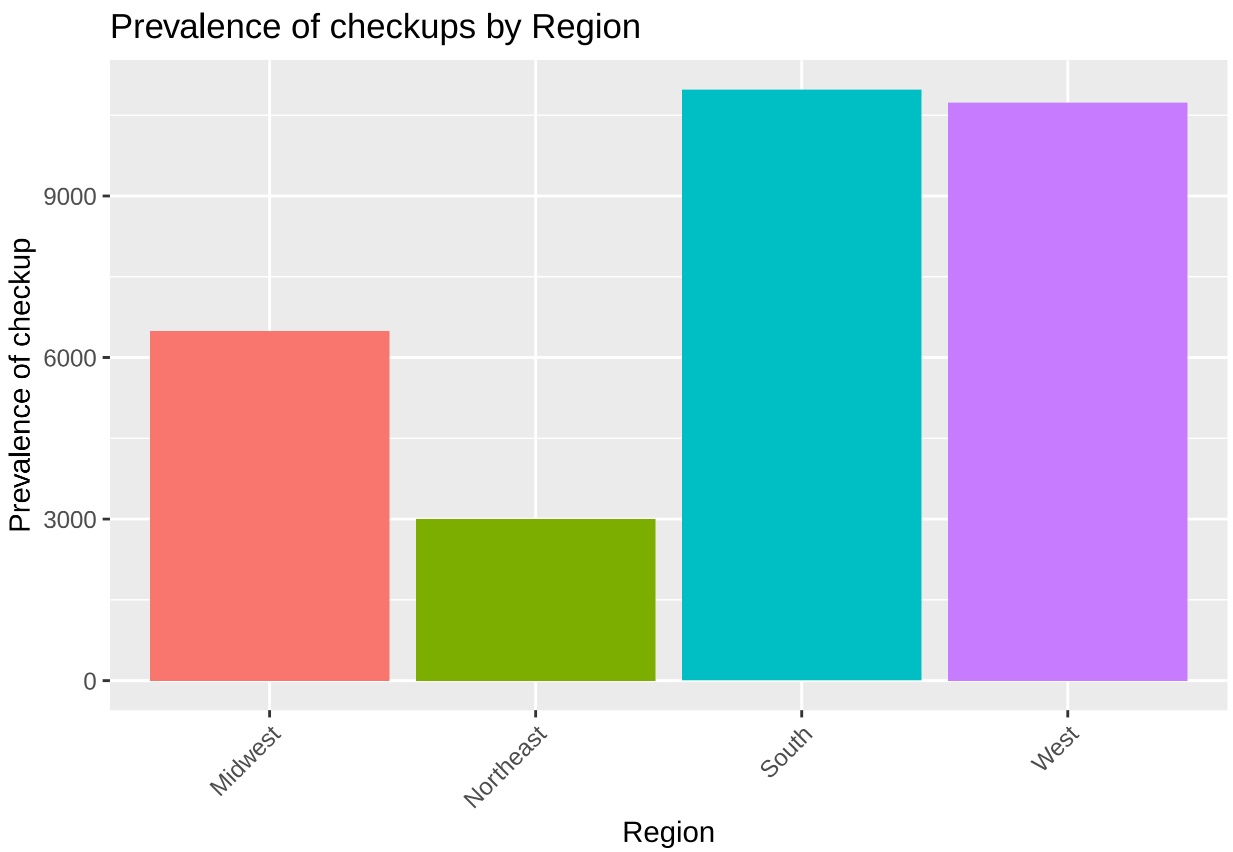
```
#madeleine's graph for health access vs checkup
ggplot(cities, aes(x = health_access,
                   y= checkup))+
  geom_point()
```


80
75
70
65
60
checkup
10
20
30
40
health_access



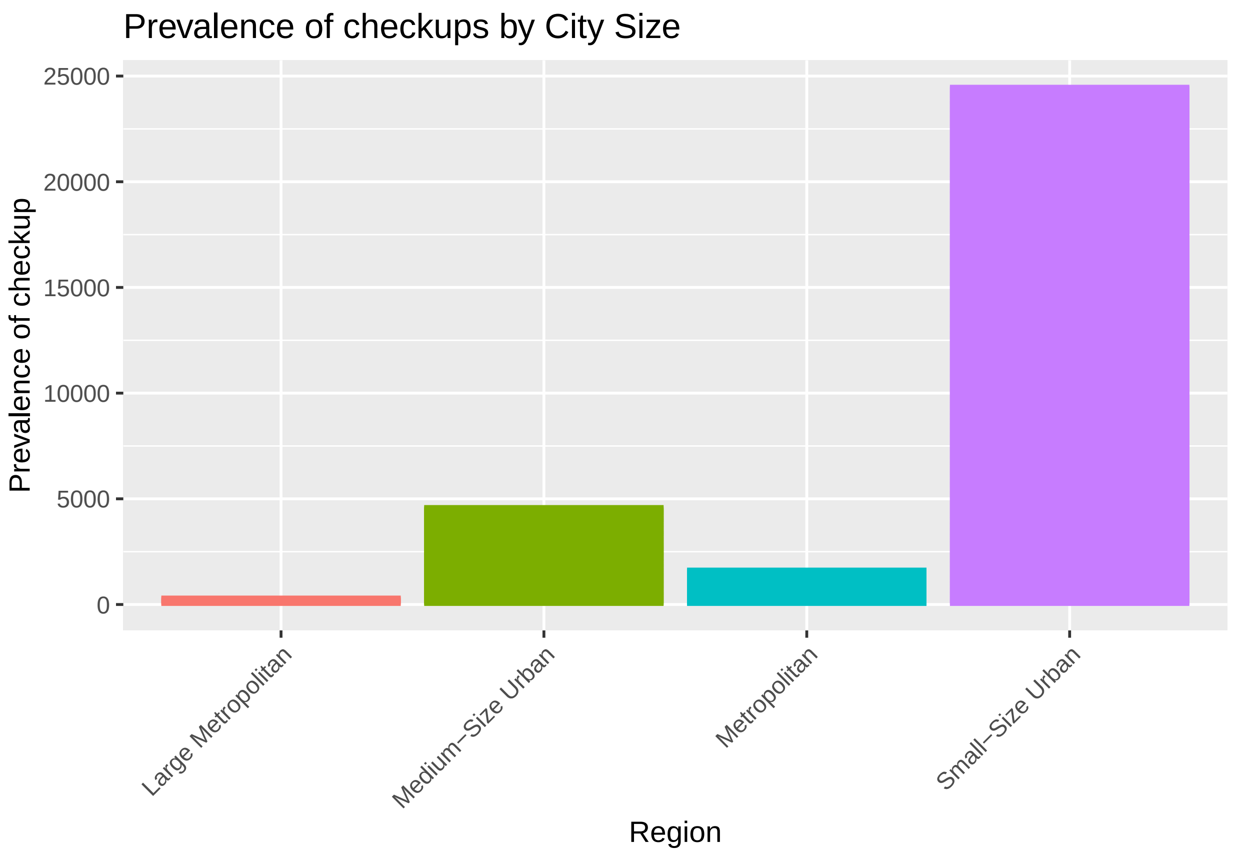
```
cities2 <- cities %>%
  filter( region != "NA")

ggplot(cities2, aes(x = region,
                    y = checkup,
                    fill = region))+
  geom_bar( stat = 'identity' )+
  labs( x = "Region",
        y = "Prevalence of checkup",
        title = "Prevalence of checkups by Region") +
  theme(axis.text.x = element_text(angle = 45,hjust=1))+
  theme(legend.position = "none")
```



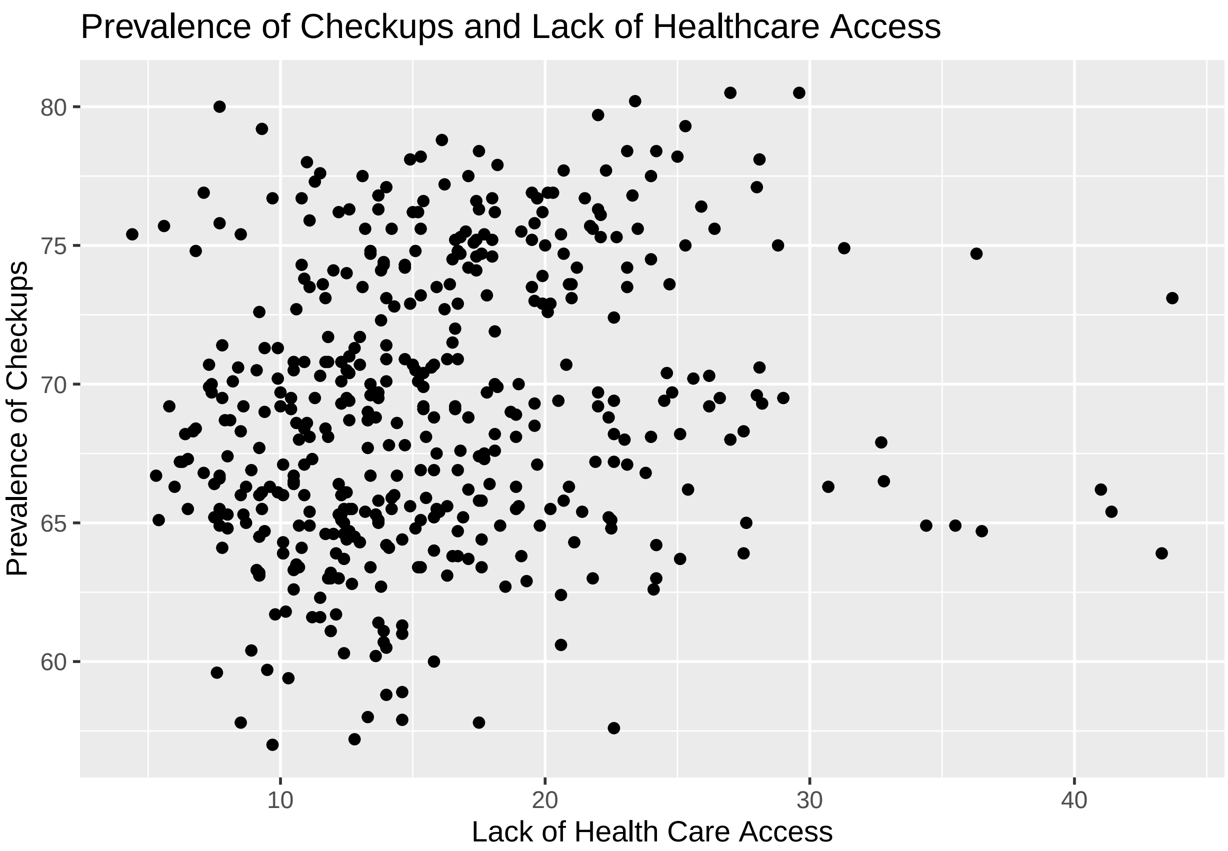
```
cities2 <- cities %>%
  filter( city_size != "NA")

ggplot(cities2, aes(x = city_size,
                    y = checkup,
                    color = city_size))+
  geom_bar( stat = 'identity' )+
  labs( x = "Region",
        y = "Prevalence of checkup",
        title = "Prevalence of checkups by City Size") +
  theme(axis.text.x = element_text(angle = 45,hjust=1))+
  theme(legend.position = "none")
```



```
#doesn't look that helpful tbh won't hurt my feelings if u trash it
```

```
ggplot(cities, aes(x = health_access,
                   y= checkup))+
  geom_point()+
  labs( x = "Lack of Health Care Access",
        y = "Prevalence of Checkups",
        title = "Prevalence of Checkups and Lack of Healthcare Access")
```



```
#cloud of points means something-- don't forget this later on
```

```
#WE PROBABLY DON'T NEEED THIS CHUNK ANYMORE I MOVED THE CORRELATION TEST TO THE BEGINNING PART

# cities_heatmap <- melt(cities, id.vars = "checkup", measure.vars =
#                          c("colon_screen", "mammogram", "pap_test"))

#head(cities_heatmap)
# ggplot(cities, aes(checkup, )) +


#correlation1 <- cor.test(cities$checkup_n, cities$pap_n_screened,
#                         method = "pearson")
#correlation2 <- cor.test(cities$checkup_n, cities$colon_n_screened,
#                         method = "pearson")
#correlation3 <- cor.test(cities$checkup_n, cities$mammogram_n_screened,
#                         method = "pearson")
#print(correlation1)
#print(correlation2)
#print(correlation3)
```

Multiple Logistic Regression Model

```
##
## Call:
## glm(formula = cbind(checkup_n, no_checkup_n) ~ region + city_size +
##     health_access, family = binomial, data = model_cities)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -99.359  -18.975   -0.627   14.963  130.327  
## 
## Coefficients:
##                               Estimate  Std. Error  z value
## (Intercept)                 0.97037012  0.00122436  792.555
## regionNortheast             0.19874789  0.00093326  212.960
## regionSouth                 0.10411063  0.00084367  123.402
## regionWest                 -0.27722312  0.00071380 -388.374
## city_sizeMedium-Size Urban  0.01336343  0.00086185   15.506
## city_sizeMetropolitan      -0.00289017  0.00084854   -3.406
## city_sizeSmall-Size Urban  -0.01782009  0.00077588  -22.968
## health_access              -0.00596421  0.00005375 -110.965
##                                        Pr(>|z|)    
## (Intercept)                < 0.0000000000000002 ***
## regionNortheast            < 0.0000000000000002 ***
## regionSouth                < 0.0000000000000002 ***
## regionWest                 < 0.0000000000000002 ***
## city_sizeMedium-Size Urban < 0.0000000000000002 ***
## city_sizeMetropolitan                  0.000659 ***
## city_sizeSmall-Size Urban  < 0.0000000000000002 ***
## health_access              < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 808649  on 336  degrees of freedom
## Residual deviance: 290529  on 329  degrees of freedom
## AIC: 294628
## 
## Number of Fisher Scoring iterations: 3
```

```
## 
## Call:
## glm(formula = cbind(checkup_n, no_checkup_n) ~ region + city_size + 
##     health_access + region * city_size + region * health_access + 
##     city_size * health_access, family = binomial, data = model_cities)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -99.888  -17.030    1.229   14.169   82.060  
## 
## Coefficients: (1 not defined because of singularities)
##                                              Estimate Std. Error z value
## (Intercept)                                 0.8623439  0.0029286 294.451
## regionNortheast                             0.1880943  0.0043932  42.815
## regionSouth                                 0.2746490  0.0041054  66.900
## regionWest                                 -0.1962290  0.0035729 -54.922
## city_sizeMedium-Size Urban                 -0.0457458  0.0027110 -16.874
## city_sizeMetropolitan                       0.0473568  0.0030403  15.576
## city_sizeSmall-Size Urban                  -0.0424727  0.0017036 -24.931
## health_access                               0.0017293  0.0001597  10.827
## regionNortheast:city_sizeMedium-Size Urban  0.1512818  0.0032379  46.723
## regionSouth:city_sizeMedium-Size Urban      0.1308588  0.0028905  45.273
## regionWest:city_sizeMedium-Size Urban      -0.0256434  0.0023203 -11.052
## regionNortheast:city_sizeMetropolitan       0.1545007  0.0034272  45.081
## regionSouth:city_sizeMetropolitan          -0.1039155  0.0029578 -35.132
## regionWest:city_sizeMetropolitan           -0.1806135  0.0024790 -72.858
## regionNortheast:city_sizeSmall-Size Urban   0.0338259  0.0024356  13.888
## regionSouth:city_sizeSmall-Size Urban       0.1124774  0.0024916  45.143
## regionWest:city_sizeSmall-Size Urban       -0.0428439  0.0021516 -19.912
## regionNortheast:health_access              -0.0013245  0.0002692  -4.921
## regionSouth:health_access                  -0.0131765  0.0001754 -75.137
## regionWest:health_access                   -0.0013723  0.0001888  -7.268
## city_sizeMedium-Size Urban:health_access    0.0002466  0.0001376   1.792
## city_sizeMetropolitan:health_access         0.0033393  0.0001517  22.013
## city_sizeSmall-Size Urban:health_access            NA         NA      NA
##                                                        Pr(>|z|)    
## (Intercept)                                < 0.0000000000000002 ***
## regionNortheast                            < 0.0000000000000002 ***
## regionSouth                                < 0.0000000000000002 ***
## regionWest                                 < 0.0000000000000002 ***
## city_sizeMedium-Size Urban                 < 0.0000000000000002 ***
## city_sizeMetropolitan                      < 0.0000000000000002 ***
## city_sizeSmall-Size Urban                  < 0.0000000000000002 ***
## health_access                              < 0.0000000000000002 ***
## regionNortheast:city_sizeMedium-Size Urban < 0.0000000000000002 ***
## regionSouth:city_sizeMedium-Size Urban     < 0.0000000000000002 ***
## regionWest:city_sizeMedium-Size Urban      < 0.0000000000000002 ***
## regionNortheast:city_sizeMetropolitan      < 0.0000000000000002 ***
## regionSouth:city_sizeMetropolitan          < 0.0000000000000002 ***
## regionWest:city_sizeMetropolitan           < 0.0000000000000002 ***
## regionNortheast:city_sizeSmall-Size Urban  < 0.0000000000000002 ***
## regionSouth:city_sizeSmall-Size Urban      < 0.0000000000000002 ***
## regionWest:city_sizeSmall-Size Urban       < 0.0000000000000002 ***
## regionNortheast:health_access                 0.000000861882865 ***
## regionSouth:health_access                  < 0.0000000000000002 ***
## regionWest:health_access                      0.000000000000366 ***
## city_sizeMedium-Size Urban:health_access                 0.0732 .  
## city_sizeMetropolitan:health_access        < 0.0000000000000002 ***
## city_sizeSmall-Size Urban:health_access                      NA    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 808649  on 336  degrees of freedom
## Residual deviance: 241700  on 315  degrees of freedom
## AIC: 245827
## 
## Number of Fisher Scoring iterations: 3
```

Model Validation

model that was made from the training data set compared to the rest of the test data points from the cities data

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# predictions from no interaction model

test_prob1 = predict(checkup_fit, newdata = cities_test, type = "response")
print(test_prob1)
```

```
##         1         2         3         4         5         6         7         8
## 0.6455606 0.7314895 0.7011851 0.7228949 0.7112079 0.7050252 0.7078892 0.6422785
##         9        10        11        12        13        14        15        16
## 0.6523540 0.7276781 0.6500896 0.7057882 0.7093670 0.6459699 0.6957643 0.7177244
##        17        18        19        20        21        22        23        24
## 0.7083823 0.7351469 0.7208554 0.7154229 0.7243224 0.6459699 0.7294139 0.6503223
##        25        26        27        28        29        30        31        32
## 0.7157870 0.7352215 0.6481492 0.7284711 0.7143288 0.7110314 0.6446047 0.6527981
##        33        34        35        36        37        38        39        40
## 0.7096128 0.7054165 0.7094899 0.7451987 0.7226520 0.6461062 0.7220539 0.7156657
##        41        42        43        44        45        46        47        48
## 0.7207354 0.7451987 0.7174826 0.7044241 0.7430410 0.7104724 0.6389831 0.7072722
##        49        50        51        52        53        54        55        56
## 0.7123092 0.6467878 0.6435109 0.7165145 0.7284711 0.6454241 0.7445186 0.7242033
##        57        58        59        60        61        62        63        64
## 0.7335473 0.6511356 0.7117661 0.7243224 0.7476684 0.7315275 0.7064071 0.6512711
##        65        66        67        68        69        70        71        72
## 0.6370547 0.7013101 0.6485571 0.7217288 0.7072722 0.6484598 0.7081709 0.6439212
##        73        74        75        76        77        78        79        80
## 0.6297147 0.6896344 0.6297147 0.7254268 0.7297669 0.7100500 0.7293296 0.7290941
##        81        82        83        84        85        86        87        88
## 0.6609976 0.6507291 0.7171198 0.6437845 0.7104724 0.7486131 0.6499197 0.7189311
##        89        90        91        92        93        94        95        96
## 0.7278425 0.6500510 0.7045482 0.6411815 0.6967931 0.7279606 0.7207354 0.6476049
##        97        98        99       100       101       102       103       104
## 0.7131987 0.6388455 0.6420044 0.7485008 0.7246796 0.6462426 0.7173617 0.6443314
##       105       106       107       108       109       110       111       112
## 0.6432372 0.7183281 0.7304339 0.6462426 0.6435109 0.6482852 0.6330461 0.6476049
##       113       114
## 0.7081358 0.7275599
```

```
cities_test1 <-
  merge(cities_test, test_prob1, by = "row.names", all.x = TRUE)

# prediction from model with interaction
```

```
test_prob2 = predict(checkup_fit2, newdata = cities_test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
print(test_prob2)
```

```
##         1         2         3         4         5         6         7         8
## 0.6422934 0.7411420 0.7003051 0.7165060 0.6973570 0.6932856 0.6983415 0.6424903
##         9        10        11        12        13        14        15        16
## 0.6418832 0.7359407 0.6464654 0.6989605 0.6979042 0.6422688 0.6753844 0.7251276
##        17        18        19        20        21        22        23        24
## 0.6981958 0.7333170 0.7310200 0.7207714 0.7374972 0.6422688 0.7469145 0.6420063
##        25        26        27        28        29        30        31        32
## 0.7214620 0.7408934 0.6421376 0.7451795 0.7186932 0.7124025 0.6423508 0.6461895
##        33        34        35        36        37        38        39        40
## 0.6978313 0.6990696 0.6978677 0.7402167 0.7343828 0.6422606 0.7332648 0.7212319
##        41        42        43        44        45        46        47        48
## 0.7307948 0.7402167 0.7246711 0.6993606 0.7403646 0.6975759 0.6426870 0.6985237
##        49        50        51        52        53        54        55        56
## 0.6970284 0.6422196 0.6424164 0.7228401 0.7451795 0.6423016 0.7402634 0.7372755
##        57        58        59        60        61        62        63        64
## 0.7466791 0.6419571 0.7138077 0.7374972 0.7419380 0.7507897 0.6987785 0.6419489
##        65        66        67        68        69        70        71        72
## 0.6428018 0.7002688 0.6421130 0.7249147 0.6985237 0.6466309 0.7003166 0.6423918
##        73        74        75        76        77        78        79        80
## 0.6432361 0.6706960 0.6432361 0.7317844 0.7475631 0.7105226 0.7389767 0.7385443
##        81        82        83        84        85        86        87        88
## 0.6453472 0.6419817 0.7239854 0.6424000 0.6975759 0.7624107 0.6404261 0.7274034
##        89        90        91        92        93        94        95        96
## 0.7413828 0.6420227 0.6993242 0.6425559 0.7015739 0.7413750 0.7307948 0.6421704
##        97        98        99       100       101       102       103       104
## 0.6986136 0.6426952 0.6425067 0.7624225 0.7381615 0.6422524 0.7244426 0.6423672
##       105       106       107       108       109       110       111       112
## 0.6424329 0.7262670 0.7412119 0.6422524 0.6424164 0.6421294 0.6430395 0.6421704
##       113       114
## 0.6982687 0.7357229
```

```
cities_test2 <-
  merge(cities_test, test_prob2, by = "row.names", all.x = TRUE)

#rmse against model/training data set

test_prob_model = predict(checkup_fit, newdata = model_cities, type = "response")
model_cities1 <- merge(model_cities, test_prob_model, by = "row.names", all.x = TRUE)
rmse(model_cities1, truth = (checkup/100), estimate = test_prob_model)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0614
```

```
test_prob_model2 = predict(checkup_fit2, newdata = model_cities, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
model_cities2 <- merge(model_cities, test_prob_model2, by = "row.names", all.x = TRUE)
rmse(model_cities2, truth = (checkup/100), estimate = test_prob_model)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0614
```

```
#calculate root mean standard error of both models against test data set

rmse(cities_test1, truth = (checkup/100), estimate = test_prob1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0612
```

```
rmse(cities_test2, truth = (checkup/100), estimate = test_prob2)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      0.0641
```