Disclaimer: It is probably easier to read this assignment as `.Rmd` and not as PDF, because there are a lot of outputs and plots, which can get a little overwhelming when not viewed through RStudio.

Exercise 1

In this exercise we use the College data set from the ISLR package. We predict the number of applications received using the other variables.

```
library(ISLR)
library(tidymodels)

library(tidymodels)
library(funModeling)
library(ISLR)
library(vip)
library(forcats)
library(GGally)

?ISLR::College
```

```
College <- tibble(College)
College
```

```
## # A tibble: 777 x 18
##    Private  Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad
##    <fct>   <dbl>  <dbl>  <dbl>     <dbl>     <dbl>       <dbl>       <dbl>
##  1 Yes      1660   1232    721        23        52        2885         537
##  2 Yes      2186   1924    512        16        29        2683        1227
##  3 Yes      1428   1097    336        22        50        1036          99
##  4 Yes       417    349    137        60        89         510          63
##  5 Yes       193    146     55        16        44         249         869
##  6 Yes       587    479    158        38        62         678          41
##  7 Yes       353    340    103        17        45         416         230
##  8 Yes      1899   1720    489        37        68        1594          32
##  9 Yes      1038    839    227        30        63         973         306
## 10 Yes       582    498    172        21        44         799          78
## # ... with 767 more rows, and 10 more variables: Outstate <dbl>,
## #   Room.Board <dbl>, Books <dbl>, Personal <dbl>, PhD <dbl>, Terminal <dbl>,
## #   S.F.Ratio <dbl>, perc.alumni <dbl>, Expend <dbl>, Grad.Rate <dbl>
```

```
basic_eda <- function(data) {
  glimpse(data)
  print(status(data))
  freq(data)
  print(profiling_num(data))
  plot_num(data)
  describe(data)
}

basic_eda(College)
```

```
## Rows: 777
## Columns: 18
## $ Private     <fct> Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes~
## $ Apps        <dbl> 1660, 2186, 1428, 417, 193, 587, 353, 1899, 1038, 582, 173~
```

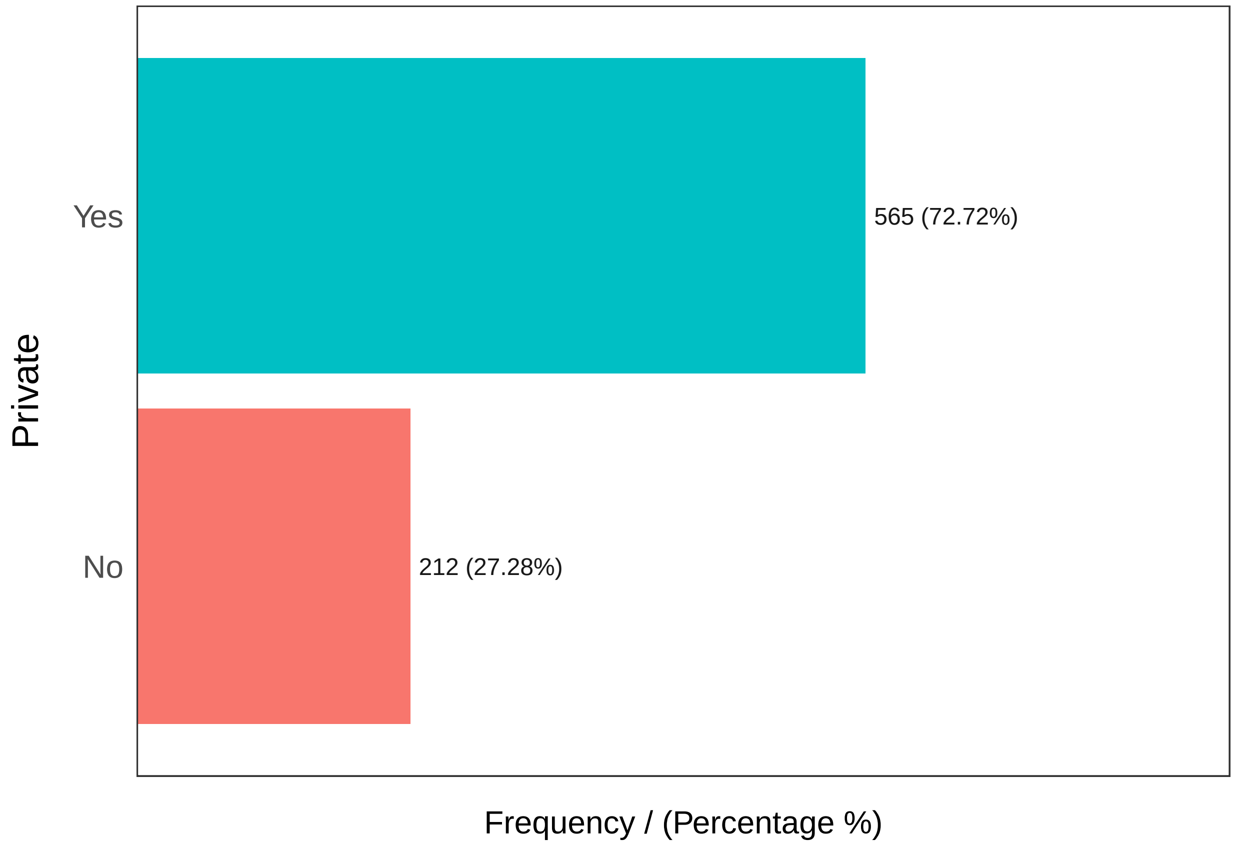
```
## $ Accept      <dbl> 1232, 1924, 1097, 349, 146, 479, 340, 1720, 839, 498, 1425~
## $ Enroll      <dbl> 721, 512, 336, 137, 55, 158, 103, 489, 227, 172, 472, 484,~
## $ Top10perc   <dbl> 23, 16, 22, 60, 16, 38, 17, 37, 30, 21, 37, 44, 38, 44, 23~
## $ Top25perc   <dbl> 52, 29, 50, 89, 44, 62, 45, 68, 63, 44, 75, 77, 64, 73, 46~
## $ F.Undergrad <dbl> 2885, 2683, 1036, 510, 249, 678, 416, 1594, 973, 799, 1830~
## $ P.Undergrad <dbl> 537, 1227, 99, 63, 869, 41, 230, 32, 306, 78, 110, 44, 638~
## $ Outstate    <dbl> 7440, 12280, 11250, 12960, 7560, 13500, 13290, 13868, 1559~
## $ Room.Board  <dbl> 3300, 6450, 3750, 5450, 4120, 3335, 5720, 4826, 4400, 3380~
## $ Books       <dbl> 450, 750, 400, 450, 800, 500, 500, 450, 300, 660, 500, 400~
## $ Personal    <dbl> 2200, 1500, 1165, 875, 1500, 675, 1500, 850, 500, 1800, 60~
## $ PhD         <dbl> 70, 29, 53, 92, 76, 67, 90, 89, 79, 40, 82, 73, 60, 79, 36~
## $ Terminal    <dbl> 78, 30, 66, 97, 72, 73, 93, 100, 84, 41, 88, 91, 84, 87, 6~
## $ S.F.Ratio   <dbl> 18.1, 12.2, 12.9, 7.7, 11.9, 9.4, 11.5, 13.7, 11.3, 11.5, ~
## $ perc.alumni <dbl> 12, 16, 30, 37, 2, 11, 26, 37, 23, 15, 31, 41, 21, 32, 26,~
## $ Expend      <dbl> 7041, 10527, 8735, 19016, 10922, 9727, 8861, 11487, 11644,~
## $ Grad.Rate   <dbl> 60, 56, 54, 59, 15, 55, 63, 73, 80, 52, 73, 76, 74, 68, 55~
##                variable q_zeros     p_zeros q_na p_na q_inf p_inf    type
## Private         Private       0 0.000000000    0    0     0     0  factor
## Apps               Apps       0 0.000000000    0    0     0     0 numeric
## Accept           Accept       0 0.000000000    0    0     0     0 numeric
## Enroll           Enroll       0 0.000000000    0    0     0     0 numeric
## Top10perc     Top10perc       0 0.000000000    0    0     0     0 numeric
## Top25perc     Top25perc       0 0.000000000    0    0     0     0 numeric
## F.Undergrad F.Undergrad       0 0.000000000    0    0     0     0 numeric
## P.Undergrad P.Undergrad       0 0.000000000    0    0     0     0 numeric
## Outstate       Outstate       0 0.000000000    0    0     0     0 numeric
## Room.Board   Room.Board       0 0.000000000    0    0     0     0 numeric
## Books             Books       0 0.000000000    0    0     0     0 numeric
## Personal       Personal       0 0.000000000    0    0     0     0 numeric
## PhD                 PhD       0 0.000000000    0    0     0     0 numeric
## Terminal       Terminal       0 0.000000000    0    0     0     0 numeric
## S.F.Ratio     S.F.Ratio       0 0.000000000    0    0     0     0 numeric
## perc.alumni perc.alumni       2 0.002574003    0    0     0     0 numeric
## Expend           Expend       0 0.000000000    0    0     0     0 numeric
## Grad.Rate     Grad.Rate       0 0.000000000    0    0     0     0 numeric
##             unique
## Private          2
## Apps           711
## Accept         693
## Enroll         581
## Top10perc       82
## Top25perc       89
## F.Undergrad    714
## P.Undergrad    566
## Outstate       640
## Room.Board     553
## Books          122
## Personal       294
## PhD             78
## Terminal        65
## S.F.Ratio      173
## perc.alumni     61
## Expend         744
## Grad.Rate       81
```

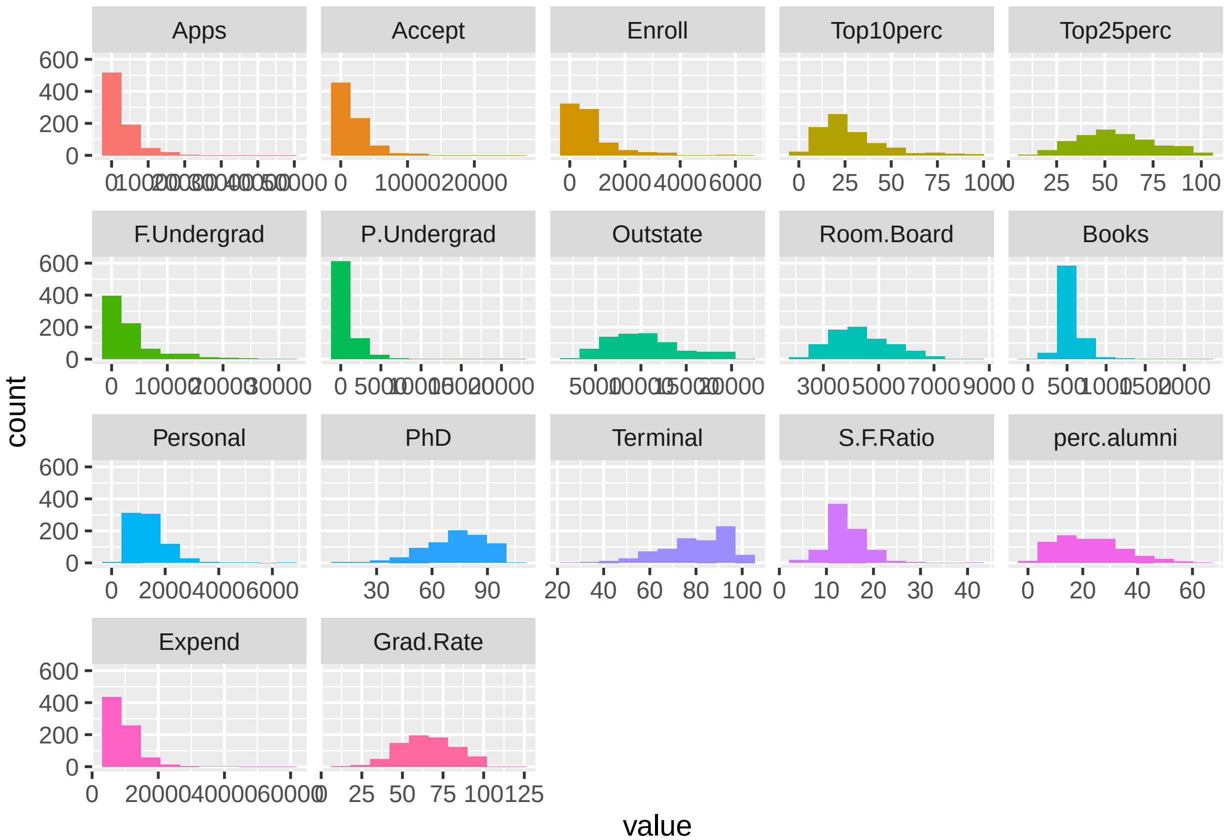
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```

```
##       variable        mean     std_dev variation_coef    p_01   p_05   p_25
## 1         Apps  3001.63835 3870.201484      1.2893630  192.52  329.8  776.0
## 2       Accept  2018.80438 2451.113971      1.2141414  146.00  272.4  604.0
## 3       Enroll   779.97297  929.176190      1.1912928   78.04  118.6  242.0
## 4    Top10perc    27.55856   17.640364      0.6401048    3.00    7.0   15.0
## 5    Top25perc    55.79665   19.804778      0.3549456   18.76   25.8   41.0
## 6  F.Undergrad  3699.90734 4850.420531      1.3109573  324.36  509.8  992.0
## 7  P.Undergrad   855.29858 1522.431887      1.7800005    4.76   20.0   95.0
## 8     Outstate 10440.66924 4023.016484      0.3853217 3737.28 4601.6 7320.0
## 9   Room.Board  4357.52638 1096.696416      0.2516787 2377.60 2735.8 3597.0
## 10       Books   549.38095  165.105360      0.3005298  250.00  350.0  470.0
## 11    Personal  1340.64221  677.071454      0.5050352  400.00  500.0  850.0
## 12         PhD    72.66023   16.328155      0.2247193   28.28   43.8   62.0
## 13    Terminal    79.70270   14.722359      0.1847159   40.04   52.8   71.0
## 14   S.F.Ratio    14.08970    3.958349      0.2809391    5.00    8.3   11.5
## 15 perc.alumni    22.74389   12.391801      0.5448410    3.00    6.0   13.0
## 16      Expend  9660.17117 5221.768440      0.5405462 3869.32 4795.8 6751.0
## 17   Grad.Rate    65.46332   17.177710      0.2624021   23.52   37.0   53.0
##      p_50    p_75    p_95     p_99   skewness  kurtosis  iqr
## 1  1558.0  3624.0 11066.2 16026.12  3.7165574 29.594559 2848
## 2  1110.0  2424.0  6979.2 11668.08  3.4111259 21.808740 1820
## 3   434.0   902.0  2757.0  4618.84  2.6852679 11.767103  660
## 4    23.0    35.0    65.2    87.48  1.4104871  5.186169   20
## 5    54.0    69.0    93.0    99.00  0.2588394  2.431791   28
## 6  1707.0  4005.0 14477.8 24540.32  2.6054157 10.639436 3013
## 7   353.0   967.0  3303.6  7477.08  5.6813582 57.673296  872
## 8  9990.0 12925.0 18498.0 19677.20  0.5082943  2.581114 5605
## 9  4200.0  5050.0  6382.0  7031.44  0.4764335  2.805940 1453
## 10  500.0   600.0   765.6  1143.00  3.4782933 31.143390  130
## 11 1200.0  1700.0  2488.8  3336.00  1.7391308 10.070544  850
## 12   75.0    85.0    95.0    99.00 -0.7666864  3.553433   23
## 13   82.0    92.0    98.0   100.00 -0.8149652  3.232752   21
## 14   13.6    16.5    21.0    23.72  0.6661462  5.537045    5
## 15   21.0    31.0    46.0    55.00  0.6057190  2.896103   18
## 16 8377.0 10830.0 17974.8 31335.48  3.4526399 21.643210 4079
## 17   65.0    78.0    94.2   100.00 -0.1135575  2.788380   25
##              range_98          range_80
## 1   [192.52, 16026.12]     [457.6, 7675]
## 2      [146, 11668.08]   [361.6, 4814.2]
## 3     [78.04, 4618.84]     [154, 1903.6]
## 4           [3, 87.48]        [10, 50.4]
## 5          [18.76, 99]        [30.6, 85]
## 6   [324.36, 24540.32]    [641, 10024.4]
## 7      [4.76, 7477.08]      [35, 2016.6]
## 8   [3737.28, 19677.2] [5568.8, 16552.8]
## 9    [2377.6, 7031.44]    [3051.2, 5950]
## 10         [250, 1143]        [400, 700]
## 11         [400, 3336]       [600, 2200]
## 12         [28.28, 99]        [50.6, 92]
## 13        [40.04, 100]          [59, 96]
## 14          [5, 23.72]        [9.9, 19.2]
```

```
## 15              [3, 55]           [8, 40]
## 16 [3869.32, 31335.48]   [5558.2, 14841]
## 17         [23.52, 100]        [44.6, 89]
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```






```
## data
##
##  18  Variables      777  Observations
## ---------------------------------------------------------------------------
## Private
##        n  missing distinct
##      777        0        2
##
## Value          No   Yes
## Frequency     212   565
## Proportion 0.273 0.727
## ---------------------------------------------------------------------------
## Apps
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      711        1     3002     3350    329.8    457.6
##      .25      .50      .75      .90      .95
##    776.0   1558.0   3624.0   7675.0  11066.2
##
## lowest :    81   100   141   150   152, highest: 19315 19873 20192 21804 48094
## ---------------------------------------------------------------------------
## Accept
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      693        1     2019     2127    272.4    361.6
##      .25      .50      .75      .90      .95
##    604.0   1110.0   2424.0   4814.2   6979.2
##
## lowest :    72    90   118   128   130, highest: 13007 13243 15096 18744 26330
```

```
## --------------------------------------------------------------------------------
## Enroll
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      581        1      780    821.5    118.6    154.0
##      .25      .50      .75      .90      .95
##    242.0    434.0    902.0   1903.6   2757.0
##
## lowest :   35   46   51   55   63, highest: 5705 5873 5874 6180 6392
## --------------------------------------------------------------------------------
## Top10perc
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0       82    0.999    27.56    18.52      7.0     10.0
##      .25      .50      .75      .90      .95
##     15.0     23.0     35.0     50.4     65.2
##
## lowest :  1  2  3  4  5, highest: 87 89 90 95 96
## --------------------------------------------------------------------------------
## Top25perc
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0       89        1     55.8    22.58     25.8     30.6
##      .25      .50      .75      .90      .95
##     41.0     54.0     69.0     85.0     93.0
##
## lowest :   9  12  13  14  16, highest:  96  97  98  99 100
## --------------------------------------------------------------------------------
## F.Undergrad
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      714        1     3700     4215    509.8    641.0
##      .25      .50      .75      .90      .95
##    992.0   1707.0   4005.0  10024.4  14477.8
##
## lowest :   139   199   201   249   282, highest: 26640 27378 28938 30017 31643
## --------------------------------------------------------------------------------
## P.Undergrad
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      566        1    855.3     1131       20       35
##      .25      .50      .75      .90      .95
##       95      353      967     2017     3304
##
## lowest :     1     2     3     4     5, highest:  9054  9310 10221 10962 21836
## --------------------------------------------------------------------------------
## Outstate
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      640        1    10441     4547     4602     5569
##      .25      .50      .75      .90      .95
##     7320     9990    12925    16553    18498
##
## lowest :  2340  2580  2700  3040  3460, highest: 19900 19960 19964 20100 21700
## --------------------------------------------------------------------------------
## Room.Board
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      553        1     4358     1236     2736     3051
##      .25      .50      .75      .90      .95
##     3597     4200     5050     5950     6382
```

```
##
## lowest : 1780 1880 1920 2146 2190, highest: 7350 7398 7400 7425 8124
## ---------------------------------------------------------------------------
## Books
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      122    0.982    549.4    152.2    350.0    400.0
##      .25      .50      .75      .90      .95
##    470.0    500.0    600.0    700.0    765.6
##
## lowest :   96  110  120  200  221, highest: 1300 1400 1495 2000 2340
## ---------------------------------------------------------------------------
## Personal
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      294    0.999     1341    710.3      500      600
##      .25      .50      .75      .90      .95
##      850     1200     1700     2200     2489
##
## lowest :  250  300  350  400  420, highest: 4110 4200 4288 4913 6800
## ---------------------------------------------------------------------------
## PhD
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0       78        1    72.66    18.18     43.8     50.6
##      .25      .50      .75      .90      .95
##     62.0     75.0     85.0     92.0     95.0
##
## lowest :   8  10  14  16  22, highest:  97  98  99 100 103
## ---------------------------------------------------------------------------
## Terminal
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0       65    0.999     79.7    16.37     52.8     59.0
##      .25      .50      .75      .90      .95
##     71.0     82.0     92.0     96.0     98.0
##
## lowest :  24  25  30  33  35, highest:  96  97  98  99 100
## ---------------------------------------------------------------------------
## S.F.Ratio
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      173        1    14.09    4.325      8.3      9.9
##      .25      .50      .75      .90      .95
##     11.5     13.6     16.5     19.2     21.0
##
## lowest :  2.5  2.9  3.3  3.9  4.3, highest: 27.2 27.6 27.8 28.8 39.8
## ---------------------------------------------------------------------------
## perc.alumni
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0       61    0.999    22.74    13.95        6        8
##      .25      .50      .75      .90      .95
##       13       21       31       40       46
##
## lowest :  0  1  2  3  4, highest: 57 58 60 63 64
## ---------------------------------------------------------------------------
## Expend
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0      744        1     9660     4650     4796     5558
```

```
##      .25      .50      .75      .90      .95
##     6751     8377    10830    14841    17975
##
## lowest :  3186  3365  3480  3605  3733, highest: 40386 41766 42926 45702 56233
## ---------------------------------------------------------------------------------
## Grad.Rate
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      777        0       81        1    65.46    19.48     37.0     44.6
##      .25      .50      .75      .90      .95
##     53.0     65.0     78.0     89.0     94.2
##
## lowest :  10  15  18  21  22, highest:  97  98  99 100 118
## ---------------------------------------------------------------------------------
```

Split the data set into a training set and a test set.

```
College_split <- initial_split(College, strata = Apps, prop = 0.5)
College_split
```

```
## <Analysis/Assess/Total>
## <388/389/777>
```

```
College_train <- training(College_split)
College_test <- testing(College_split)
```

Fit a linear model using least squares on the training set, and report the test error obtained.

```
lm_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")
```

```
lm_recipe <-
  recipe(formula = Apps ~ ., data = College_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors())
```

Note: We use the variables `Accept` and `Enroll` as independent variables here. These variables describe the number of accepted applicants and the number of new enrolled students, respectively. Depending on the application one might not know these variables when predicting the the number of applications.

```
lm_workflow <- workflow() %>%
  add_recipe(lm_recipe) %>%
  add_model(lm_spec)
```

```
lm_fit <- lm_workflow %>% fit(College_train)
```

```
augment(lm_fit, new_data = College_test) %>%
  select(Apps, .pred)
```

```
## # A tibble: 389 x 2
##     Apps  .pred
##    <dbl>  <dbl>
##  1  1660  1349.
##  2   587   625.
##  3  1038   984.
##  4   582   574.
##  5  1179  1731.
##  6  1420  1211.
```

```
## 7 1130 463.
## 8 3540 3107.
## 9 619 390.
## 10 12809 15205.
## # ... with 379 more rows
```

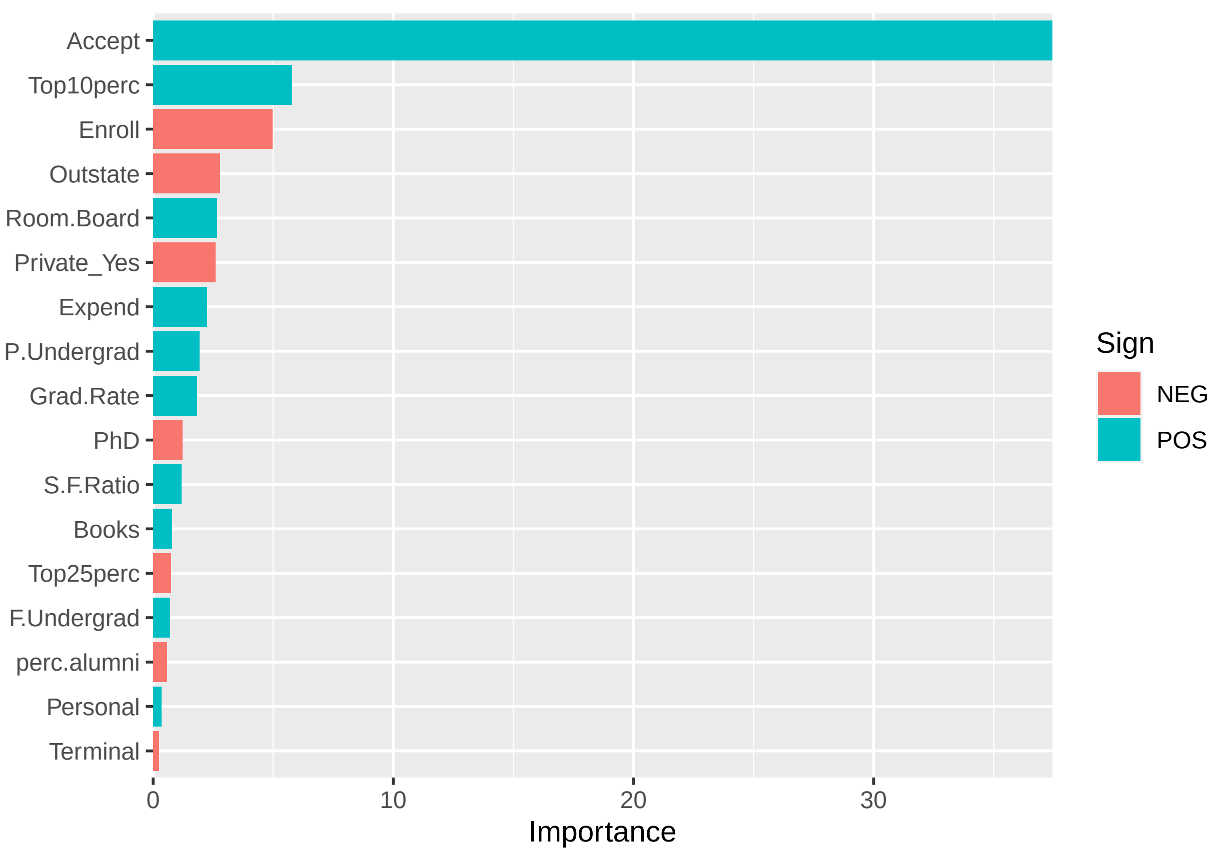
```r
augment(lm_fit, new_data = College_test) %>%
  rmse(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1183.
```

The rmse of the test error is 1183.

Out of curiosity, let's look at the feature importance.

```r
lm_fit %>%
  extract_fit_parsnip() %>%
  vi(lambda = best_penalty$penalty) %>%
  mutate(
    Importance = abs(Importance),
    Variable = fct_reorder(Variable, Importance)
  ) %>%
  ggplot(aes(x = Importance, y = Variable, fill = Sign)) +
  geom_col() +
  scale_x_continuous(expand = c(0, 0)) +
  labs(y = NULL)
```



Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
ridge_spec <- linear_reg(mixture = 0, penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

ridge_workflow <- workflow() %>%
  add_recipe(recipe = lm_recipe) %>%
  add_model(ridge_spec)
```

```
College_fold <- vfold_cv(College_train, v = 10)
College_fold
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [349/39]> Fold01
##  2 <split [349/39]> Fold02
##  3 <split [349/39]> Fold03
##  4 <split [349/39]> Fold04
##  5 <split [349/39]> Fold05
##  6 <split [349/39]> Fold06
##  7 <split [349/39]> Fold07
##  8 <split [349/39]> Fold08
##  9 <split [350/38]> Fold09
## 10 <split [350/38]> Fold10
```

```
penalty_grid <- grid_regular(
  penalty(range = c(-5, 5)), # penalty automatically uses log scale
  levels = 50
)
penalty_grid
```

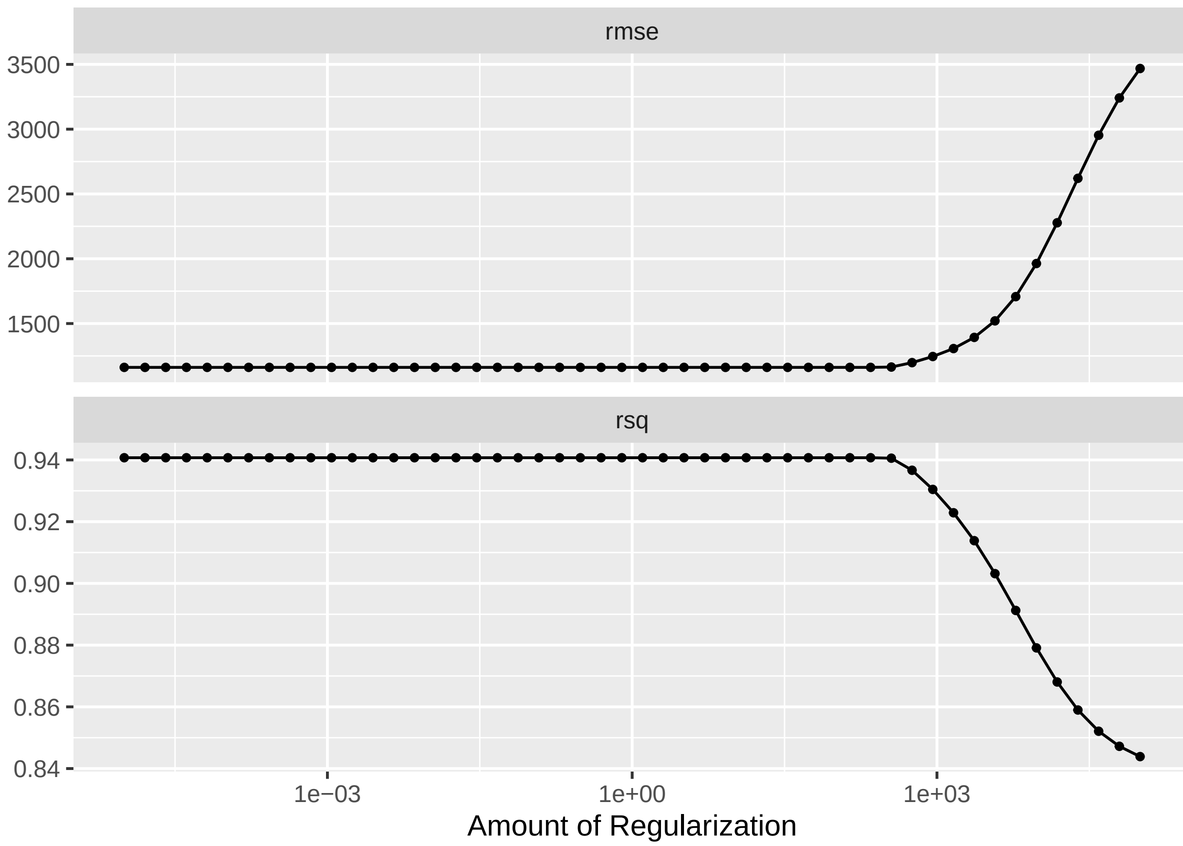
```
## # A tibble: 50 x 1
##      penalty
##        <dbl>
##  1 0.00001
##  2 0.0000160
##  3 0.0000256
##  4 0.0000409
##  5 0.0000655
##  6 0.000105
##  7 0.000168
##  8 0.000268
##  9 0.000429
## 10 0.000687
## # ... with 40 more rows
```

```
tune_res <- tune_grid(
  ridge_workflow,
  resamples = College_fold,
  grid = penalty_grid
)
```

```r
tune_res
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics           .notes
##    <list>           <chr>  <list>             <list>
##  1 <split [349/39]> Fold01 <tibble [100 x 5]> <tibble [0 x 3]>
##  2 <split [349/39]> Fold02 <tibble [100 x 5]> <tibble [0 x 3]>
##  3 <split [349/39]> Fold03 <tibble [100 x 5]> <tibble [0 x 3]>
##  4 <split [349/39]> Fold04 <tibble [100 x 5]> <tibble [0 x 3]>
##  5 <split [349/39]> Fold05 <tibble [100 x 5]> <tibble [0 x 3]>
##  6 <split [349/39]> Fold06 <tibble [100 x 5]> <tibble [0 x 3]>
##  7 <split [349/39]> Fold07 <tibble [100 x 5]> <tibble [0 x 3]>
##  8 <split [349/39]> Fold08 <tibble [100 x 5]> <tibble [0 x 3]>
##  9 <split [350/38]> Fold09 <tibble [100 x 5]> <tibble [0 x 3]>
## 10 <split [350/38]> Fold10 <tibble [100 x 5]> <tibble [0 x 3]>
```

```r
autoplot(tune_res)
```




```r
best_penalty <- select_best(tune_res, metric = "rmse")
best_penalty
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
## 1 0.00001 Preprocessor1_Model01
```

```
ridge_final <- finalize_workflow(ridge_workflow, best_penalty)
ridge_final_fit <- ridge_final %>% fit(College_train)
```

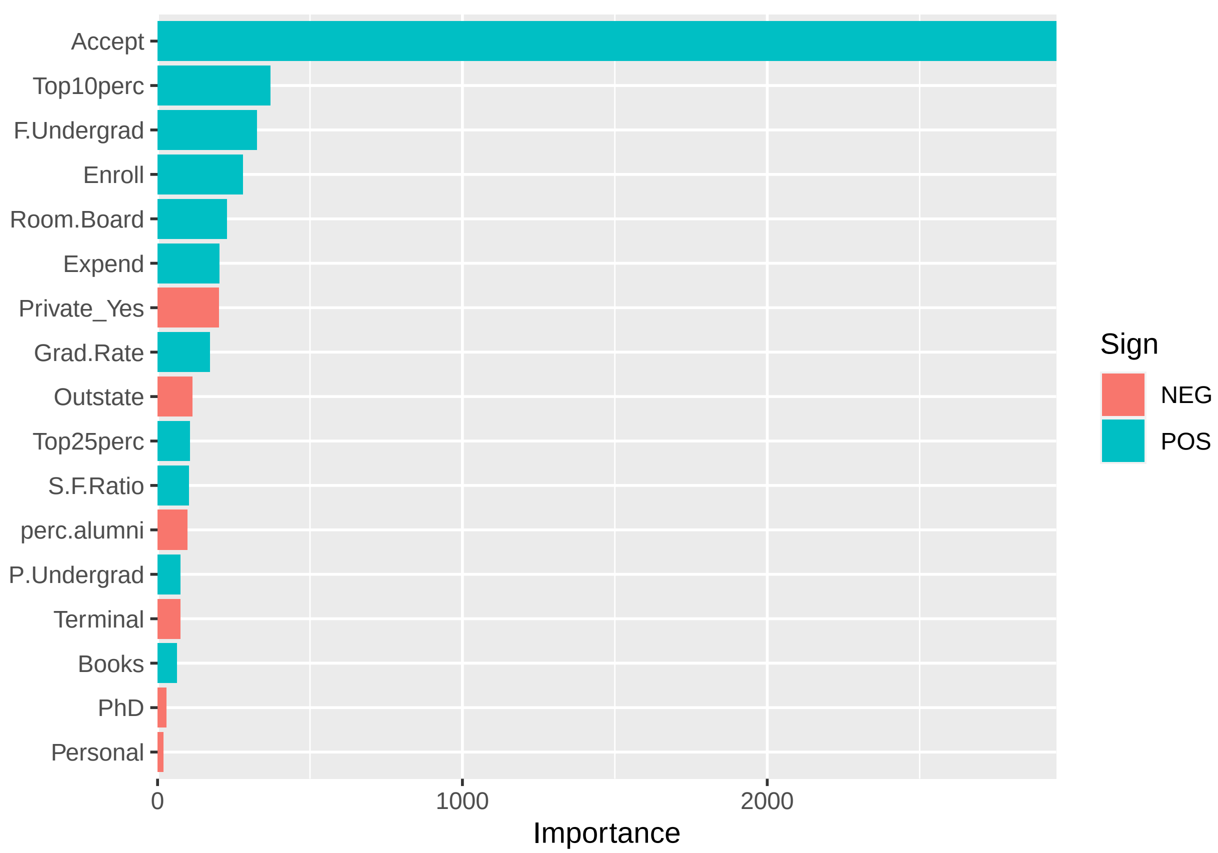
```
augment(ridge_final_fit, new_data = College_test) %>%
  rmse(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1178.
```

The test error is 1178, which is slightly lower than a linear model without regularization.

Again let's also look at the feature importance.

```
ridge_final_fit %>%
  extract_fit_parsnip() %>%
  vi(lambda = best_penalty$penalty) %>%
  mutate(
    Importance = abs(Importance),
    Variable = fct_reorder(Variable, Importance)
  ) %>%
  ggplot(aes(x = Importance, y = Variable, fill = Sign)) +
  geom_col() +
  scale_x_continuous(expand = c(0, 0)) +
  labs(y = NULL)
```



Fit a lasso model on the training set, with λ chosen by cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```r
lasso_spec <- linear_reg(mixture = 1, penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

lasso_workflow <- workflow() %>%
  add_recipe(recipe = lm_recipe) %>%
  add_model(lasso_spec)
```

```r
penalty_grid <- grid_regular(
  penalty(range = c(-5, 2)),
  levels = 50
)
```

```r
tune_res <- tune_grid(
  lasso_workflow,
  resamples = College_fold,
  grid = penalty_grid
)
tune_res
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits            id     .metrics           .notes
##    <list>            <chr>  <list>             <list>
##  1 <split [349/39]> Fold01 <tibble [100 x 5]> <tibble [0 x 3]>
##  2 <split [349/39]> Fold02 <tibble [100 x 5]> <tibble [0 x 3]>
##  3 <split [349/39]> Fold03 <tibble [100 x 5]> <tibble [0 x 3]>
##  4 <split [349/39]> Fold04 <tibble [100 x 5]> <tibble [0 x 3]>
##  5 <split [349/39]> Fold05 <tibble [100 x 5]> <tibble [0 x 3]>
##  6 <split [349/39]> Fold06 <tibble [100 x 5]> <tibble [0 x 3]>
##  7 <split [349/39]> Fold07 <tibble [100 x 5]> <tibble [0 x 3]>
##  8 <split [349/39]> Fold08 <tibble [100 x 5]> <tibble [0 x 3]>
##  9 <split [350/38]> Fold09 <tibble [100 x 5]> <tibble [0 x 3]>
## 10 <split [350/38]> Fold10 <tibble [100 x 5]> <tibble [0 x 3]>
```

```r
best_penalty <- select_best(tune_res, metric = "rmse")
best_penalty
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <chr>
## 1    26.8 Preprocessor1_Model46
```

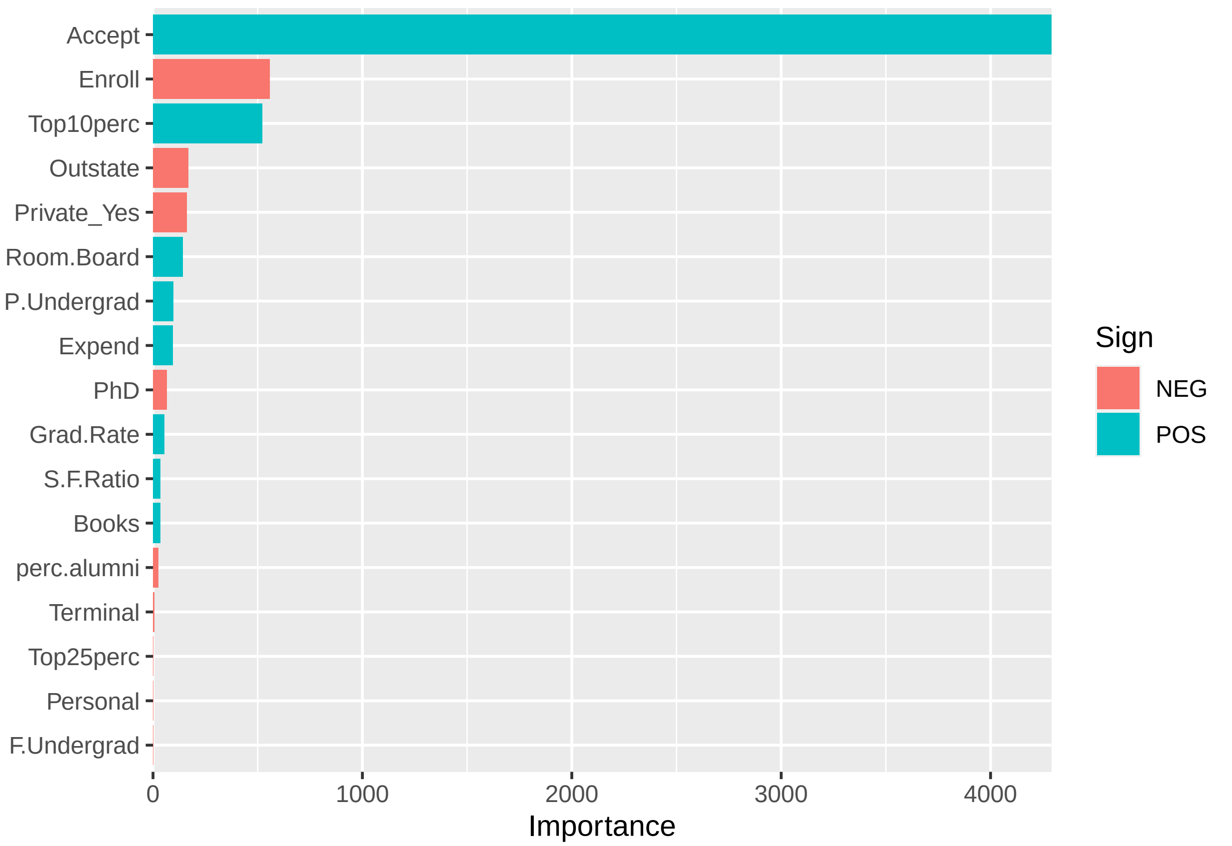
```r
lasso_final <- finalize_workflow(lasso_workflow, best_penalty)

lasso_final_fit <- fit(lasso_final, data = College_train)
```

```r
augment(ridge_final_fit, new_data = College_test) %>%
  rmse(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1178.
```

The rmse is 1178, which is the same as for Ridge Regresssion.

```
lasso_final_fit %>%
  extract_fit_parsnip() %>%
  vi(lambda = best_penalty$penalty) %>%
  mutate(
    Importance = abs(Importance),
    Variable = fct_reorder(Variable, Importance)
  ) %>%
  ggplot(aes(x = Importance, y = Variable, fill = Sign)) +
  geom_col() +
  scale_x_continuous(expand = c(0, 0)) +
  labs(y = NULL)
```



#

Exercise 2 Fit some of the non-linear models (polynomial regression, splines) discussed in the lecture to the Auto data set. Is there evidence for non-linear relationships in this data set? Create some informative plots to justify your answer.

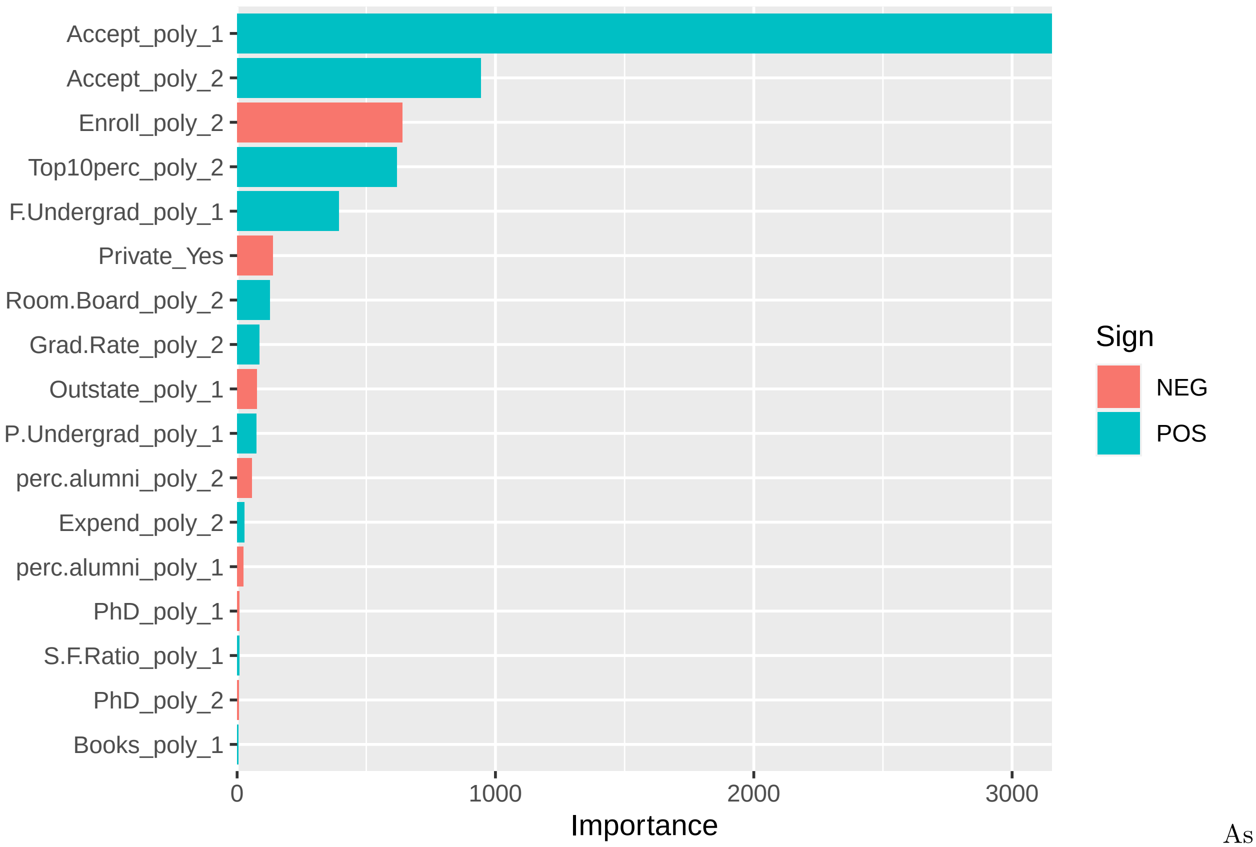
```
# Recipe
lasso_pl_recipe <-
  recipe(formula = Apps ~ ., data = College_train) %>%
  step_poly(all_numeric_predictors(), degree = 2, options = list(raw = TRUE)) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_predictors())

# Specification
lasso_pl_spec <-
```

```
  linear_reg(mixture = 1, penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

# Workflow
lasso_pl_workflow <- workflow() %>%
  add_recipe(lasso_pl_recipe) %>%
  add_model(lasso_pl_spec)

# Grid
penalty_grid <- grid_regular(
  penalty(range = c(-5, 2)),
  levels = 50
)

# Tune model
tune_res <- tune_grid(
  lasso_pl_workflow,
  resamples = College_fold,
  grid = penalty_grid
)

# Finalize model
lasso_pl_final <- finalize_workflow(lasso_pl_workflow, best_penalty)
lasso_pl_final_fit <- fit(lasso_pl_final, data = College_train)

# Check RMSE
augment(lasso_pl_final_fit, new_data = College_test) %>%
  rmse(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1083.
```

Using polynomial features reduced the error by 100, which is almost 10%. This indicates that there are non-linear relationships in the data. Let's look at the feature importance to see which polynomials are important.

```
lasso_pl_final_fit %>%
  extract_fit_parsnip() %>%
  vi(lambda = best_penalty$penalty) %>%
  filter(Importance > 2) %>%
  mutate(
    Importance = abs(Importance),
    Variable = fct_reorder(Variable, Importance)
  ) %>%
  ggplot(aes(x = Importance, y = Variable, fill = Sign)) +
  geom_col() +
  scale_x_continuous(expand = c(0, 0)) +
  labs(y = NULL)
```



As we can see, not surprisingly `Accept` is still the most important feature together with its polynomial. Let's run another polynomial (lasso) regression just with `Accept` so that we can visualize the relationship.

```
# Recipe
lasso_pl_recipe <-
  recipe(formula = Apps ~ Accept, data = College_train) %>%
  step_poly(all_numeric_predictors(), degree = 2, options = list(raw = TRUE)) %>%
  step_normalize(all_predictors())

# Specification
lasso_pl_spec <-
  linear_reg(mixture = 1, penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

# Workflow
lasso_pl_workflow <- workflow() %>%
  add_recipe(lasso_pl_recipe) %>%
  add_model(lasso_pl_spec)

# Grid
penalty_grid <- grid_regular(
  penalty(range = c(-5, 2)),
  levels = 50
)

# Tune model
tune_res <- tune_grid(
```

```
  lasso_pl_workflow,
  resamples = College_fold,
  grid = penalty_grid
)

# Finalize model
lasso_pl_final <- finalize_workflow(lasso_pl_workflow, best_penalty)
lasso_pl_final_fit <- fit(lasso_pl_final, data = College_train)

# Check RMSE
augment(lasso_pl_final_fit, new_data = College_test) %>%
  rmse(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       1439.
```

```
accept_min <- min(College$Accept)
accept_max <- max(College$Accept)
accept_range <- tibble(Accept = seq(accept_min, accept_max))
accept_range
```

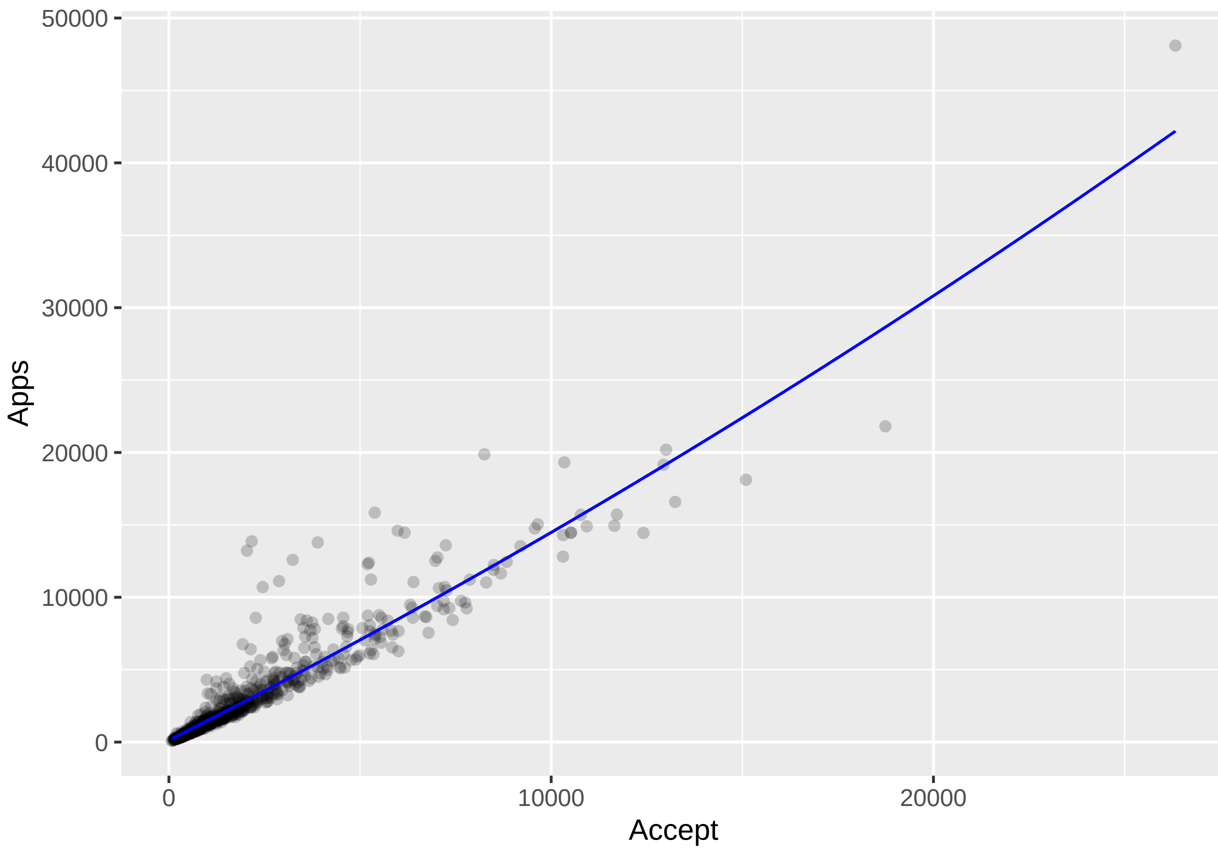
```
## # A tibble: 26,259 x 1
##    Accept
##     <int>
##  1     72
##  2     73
##  3     74
##  4     75
##  5     76
##  6     77
##  7     78
##  8     79
##  9     80
## 10     81
## # ... with 26,249 more rows
```

```
regression_lines <- bind_cols(
  predict(lasso_pl_final_fit, new_data = accept_range),
  accept_range
)

College
```

```
## # A tibble: 777 x 18
##    Private  Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad
##    <fct>   <dbl>  <dbl>  <dbl>     <dbl>     <dbl>       <dbl>       <dbl>
##  1 Yes      1660   1232    721        23        52        2885         537
##  2 Yes      2186   1924    512        16        29        2683        1227
##  3 Yes      1428   1097    336        22        50        1036          99
##  4 Yes       417    349    137        60        89         510          63
##  5 Yes       193    146     55        16        44         249         869
##  6 Yes       587    479    158        38        62         678          41
##  7 Yes       353    340    103        17        45         416         230
```

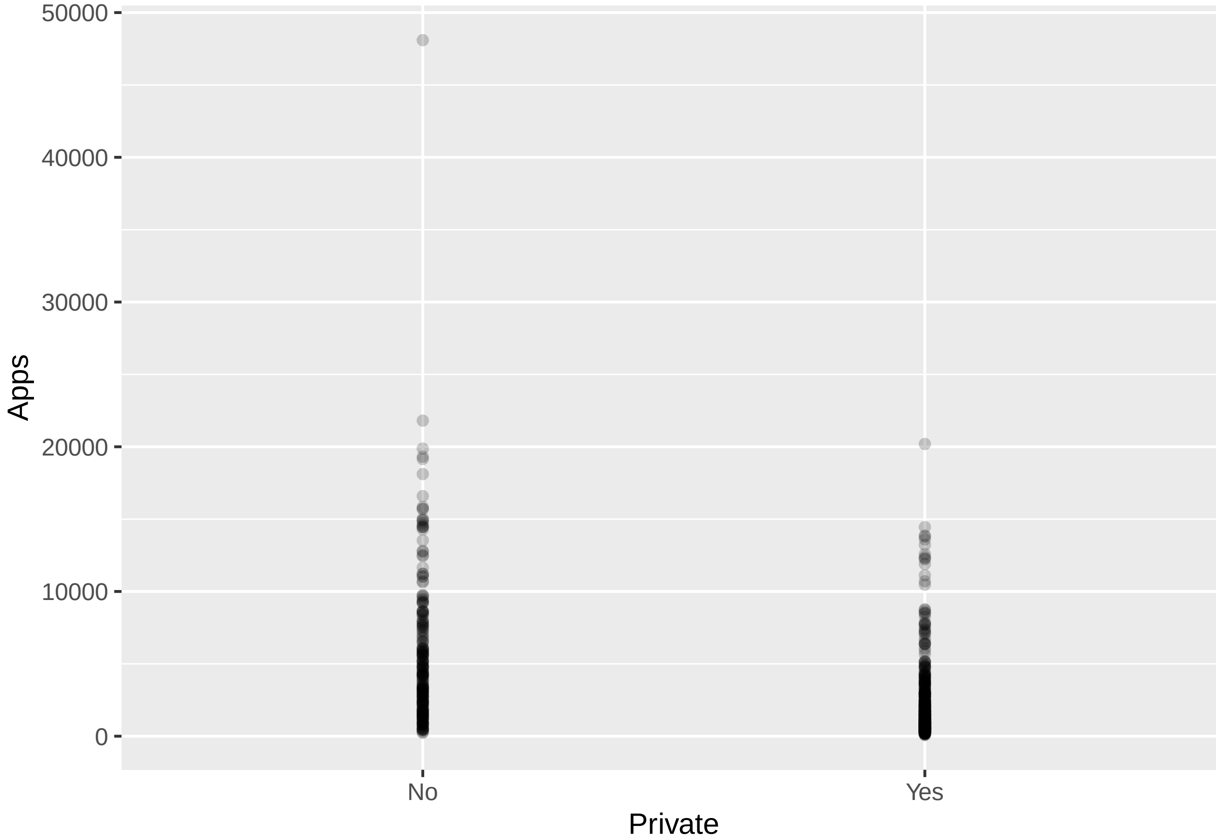
```
##  8 Yes       1899   1720    489         37         68        1594           32
##  9 Yes       1038    839    227         30         63         973          306
## 10 Yes        582    498    172         21         44         799           78
## # ... with 767 more rows, and 10 more variables: Outstate <dbl>,
## #   Room.Board <dbl>, Books <dbl>, Personal <dbl>, PhD <dbl>, Terminal <dbl>,
## #   S.F.Ratio <dbl>, perc.alumni <dbl>, Expend <dbl>, Grad.Rate <dbl>
```

```r
College %>%
  ggplot(aes(Accept, Apps)) +
  geom_point(alpha = 0.2) +
  geom_line(aes(y = .pred), data = regression_lines, color = "blue")
```

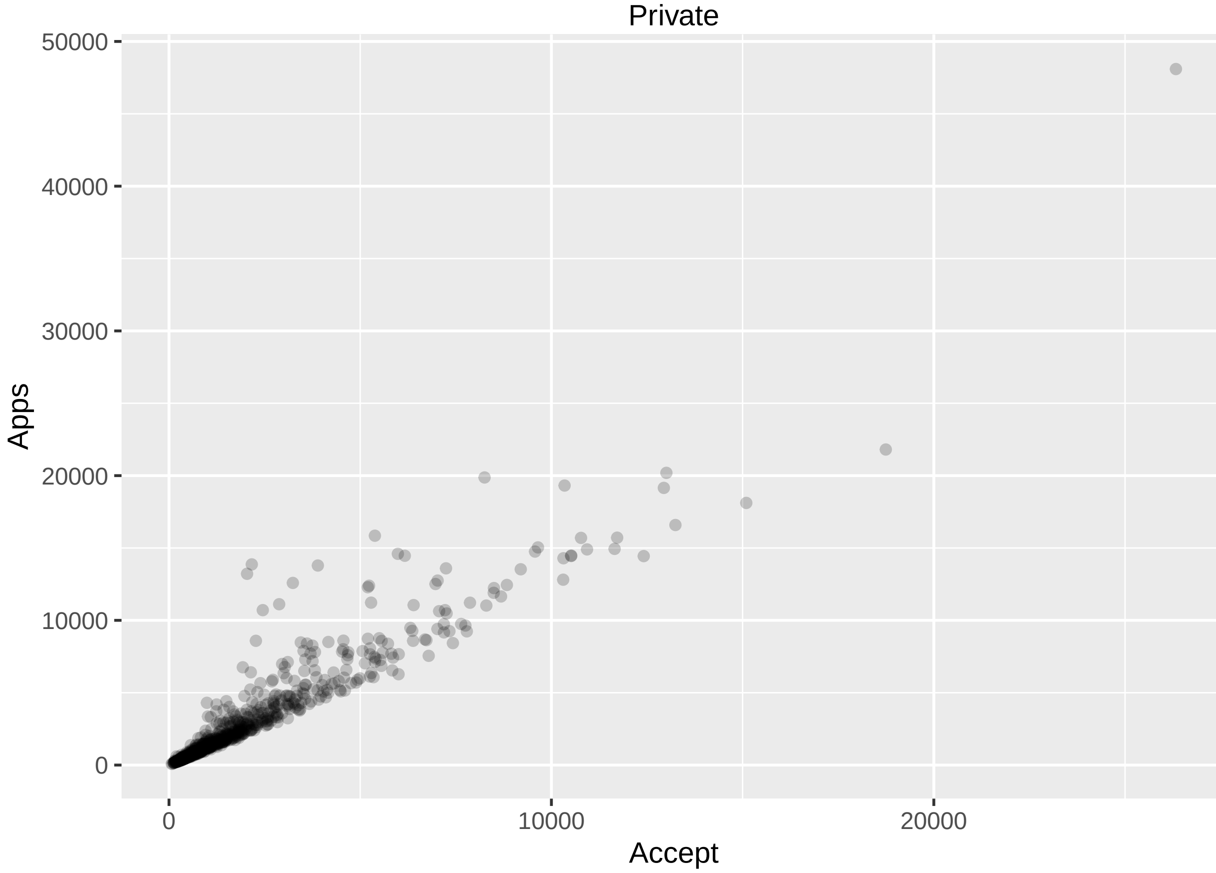


Ok this does look pretty linear. Let us take a look at the scatter plot between each variable and Accept instead.

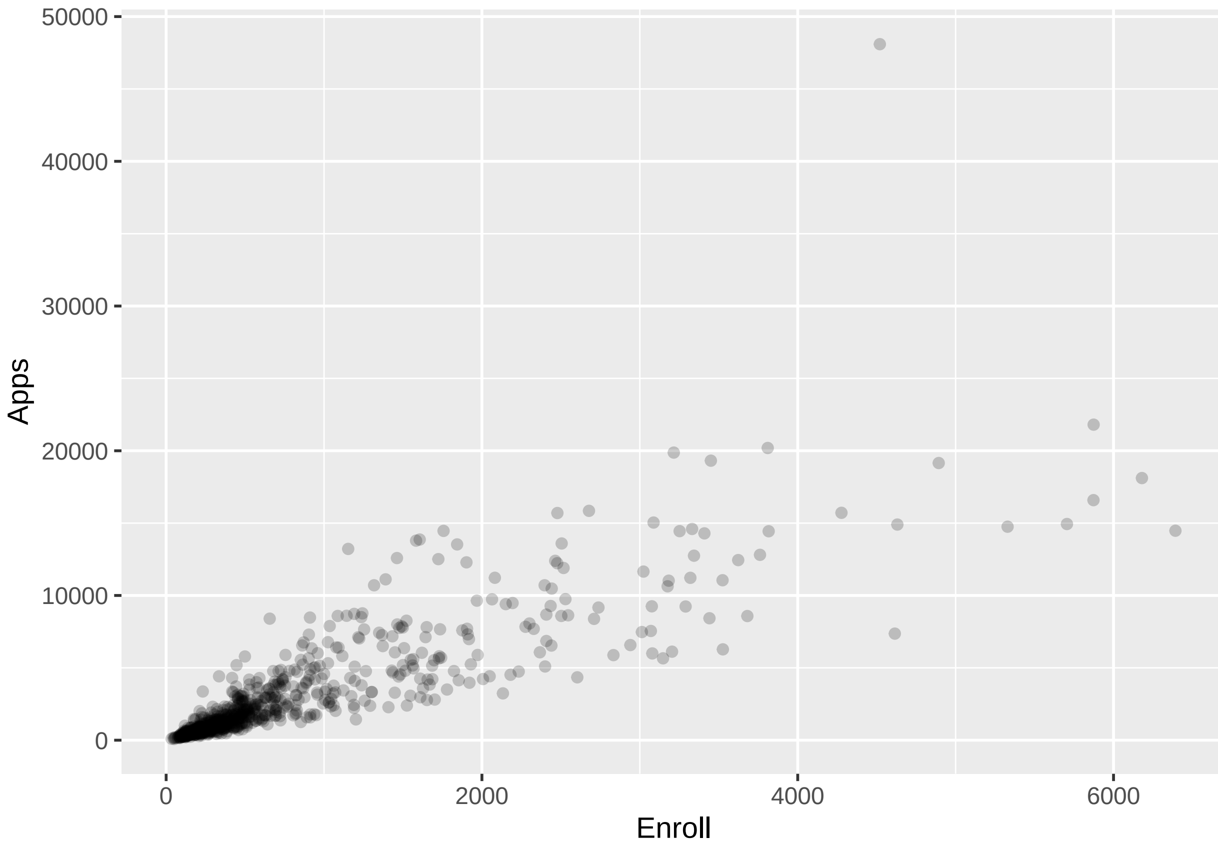
```r
# plot scatter of all variabes in College vs Apps
# get all colnames except for Apps
colnames <- colnames(College)
colnames <- colnames[colnames != "Apps"]

for (colname in colnames) {
  pl <- College %>%
    ggplot(aes_string(x = colname, y = "Apps")) +
    geom_point(alpha = 0.2)
  print(pl)
}
```



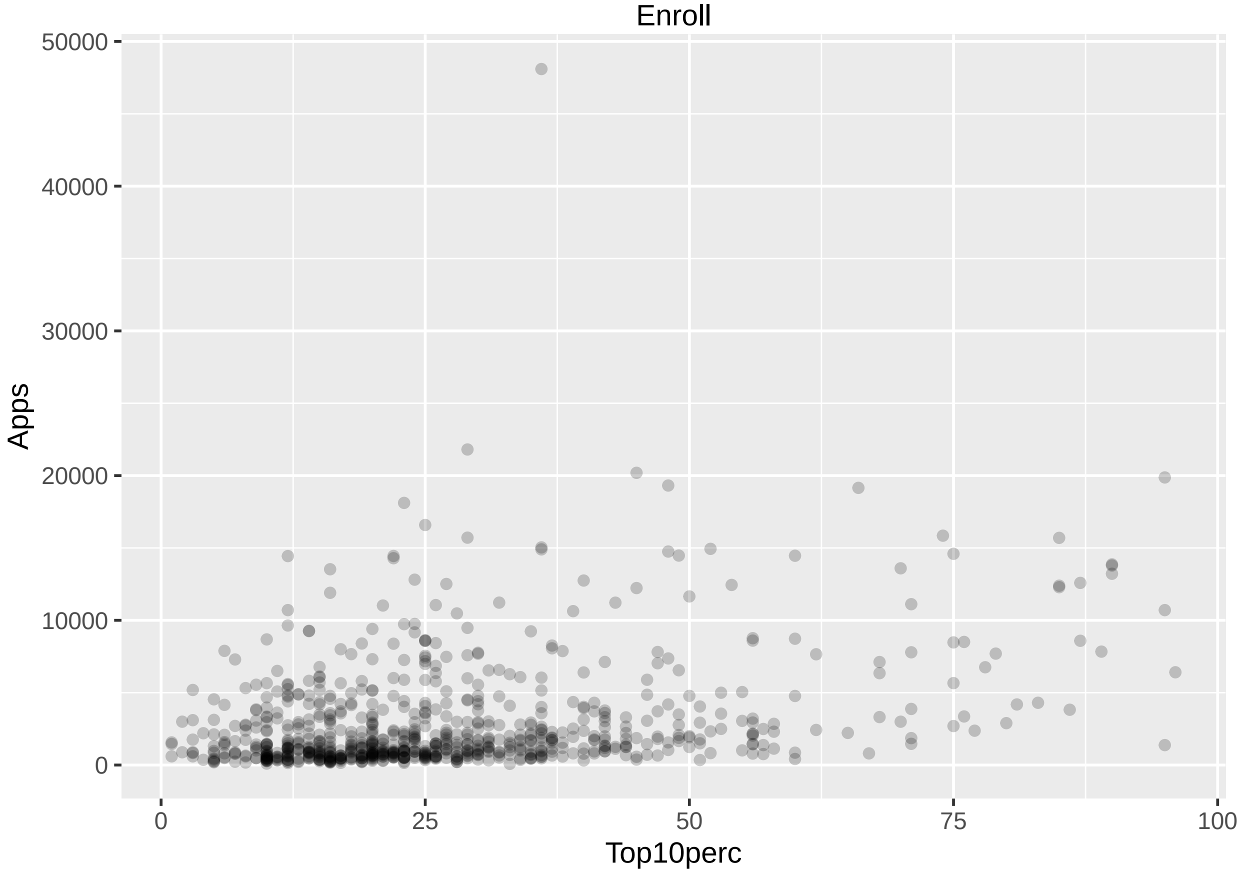
Apps
50000
40000
30000
20000
10000
0
No
Yes
Private



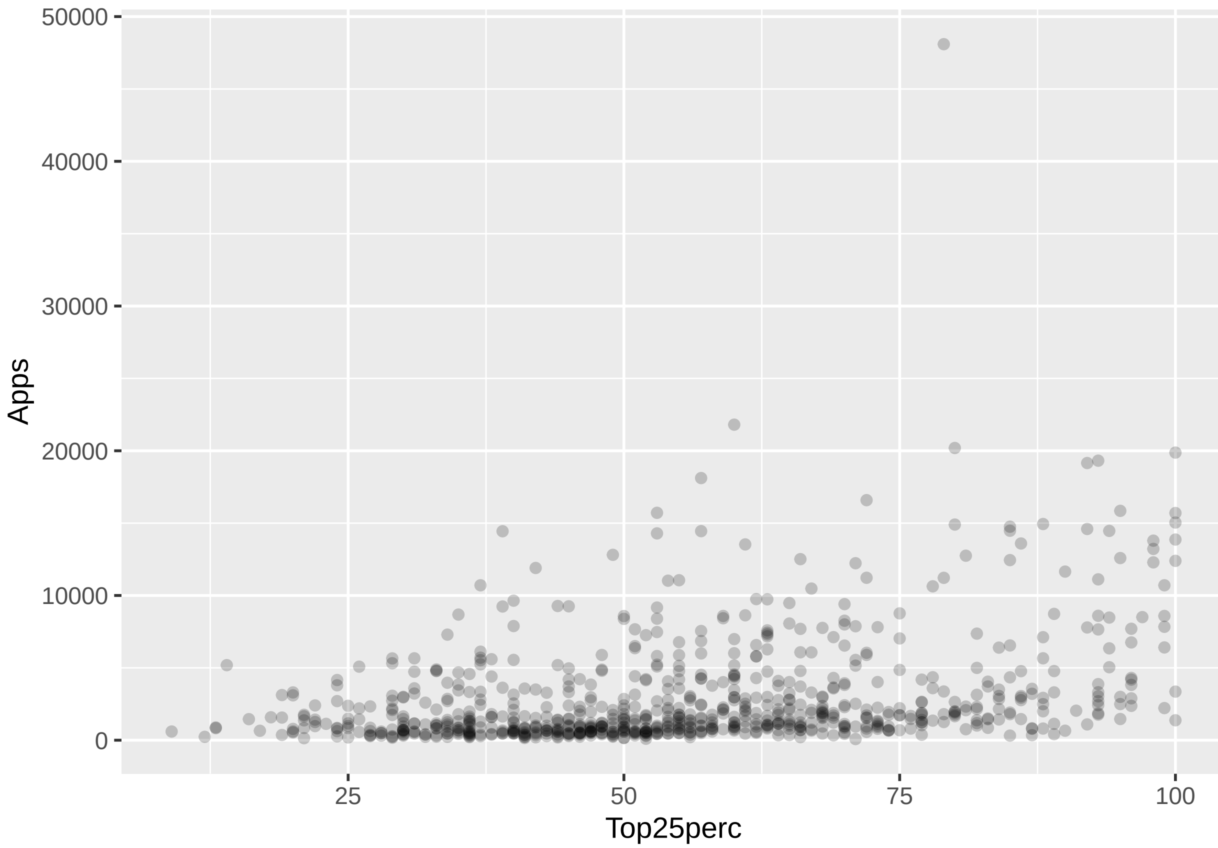
Apps
50000
40000
30000
20000
10000
0
0
10000
20000
Accept

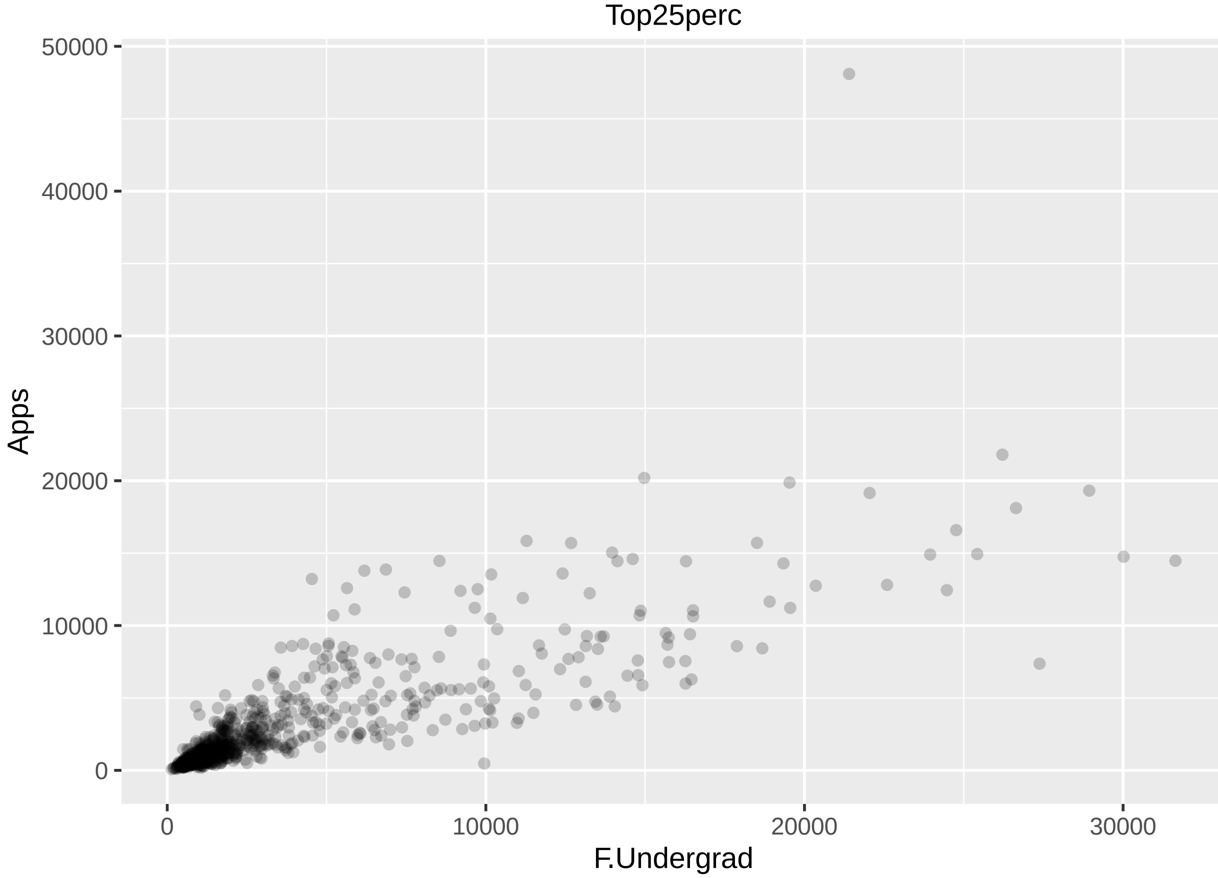


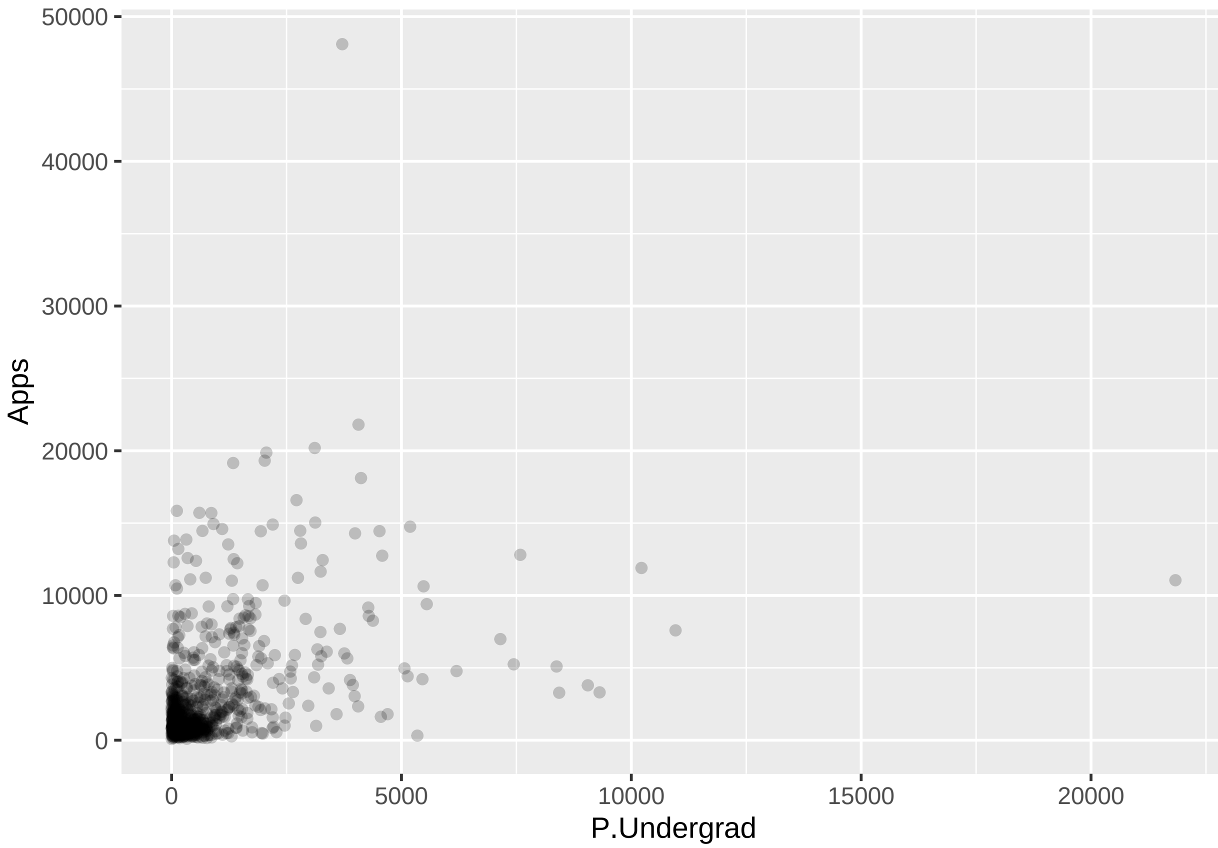
50000
40000
30000
20000
10000
0
Apps
0
2000
4000
6000
Enroll

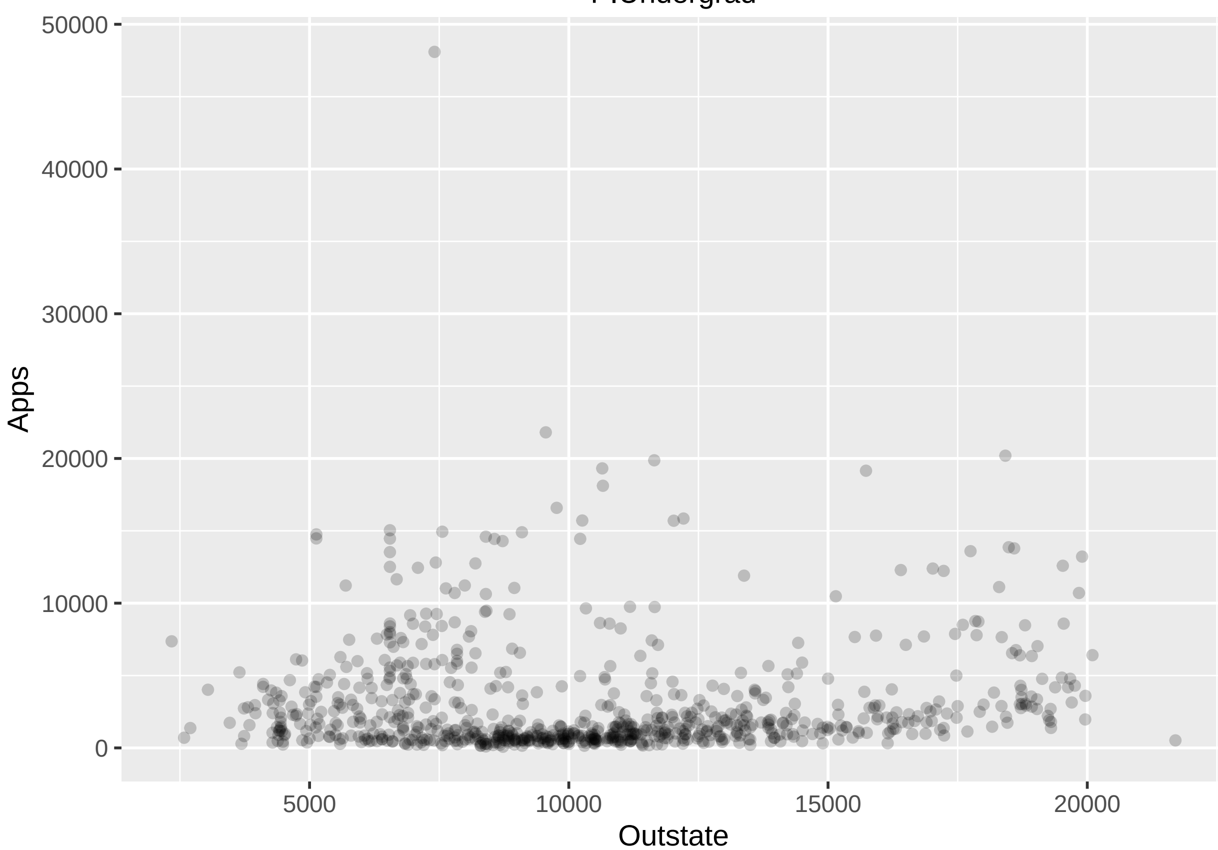


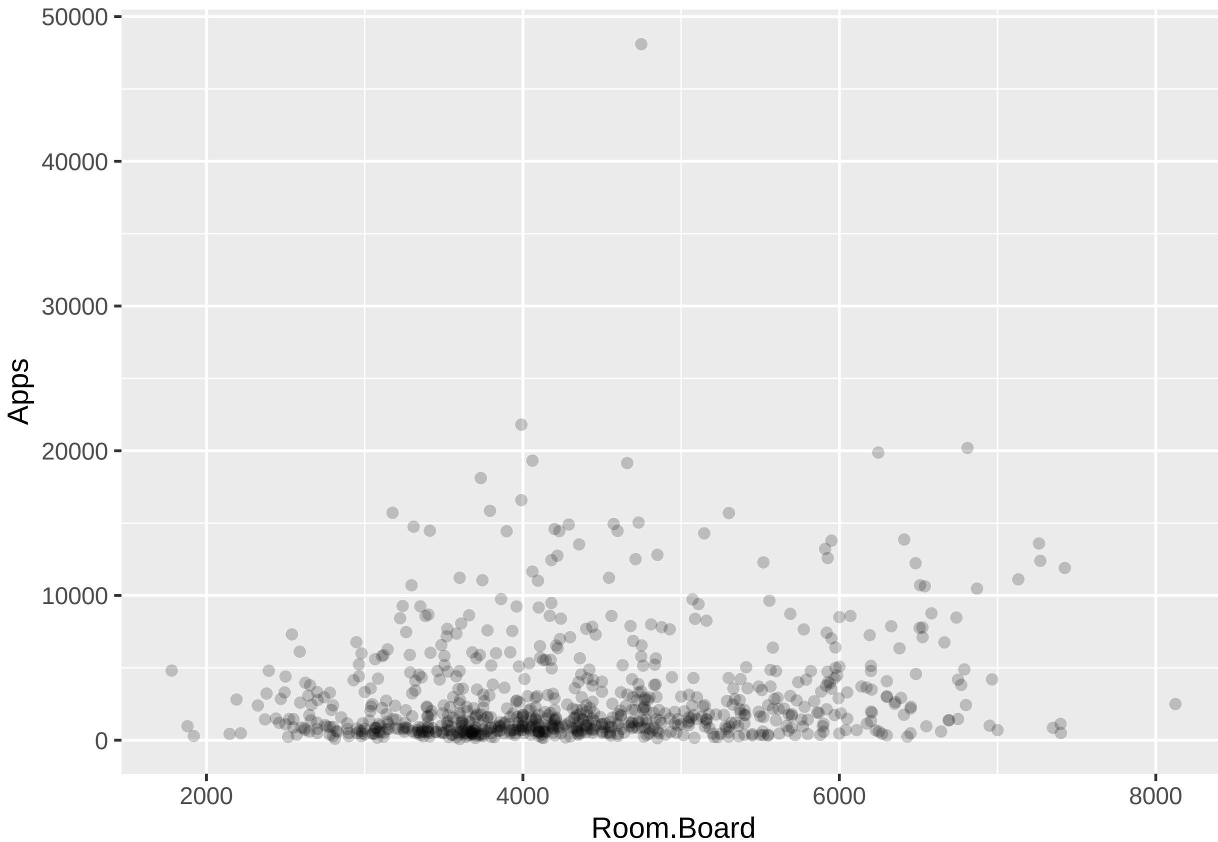
50000
40000
30000
20000
10000
0
Apps
0
25
50
75
100
Top10perc

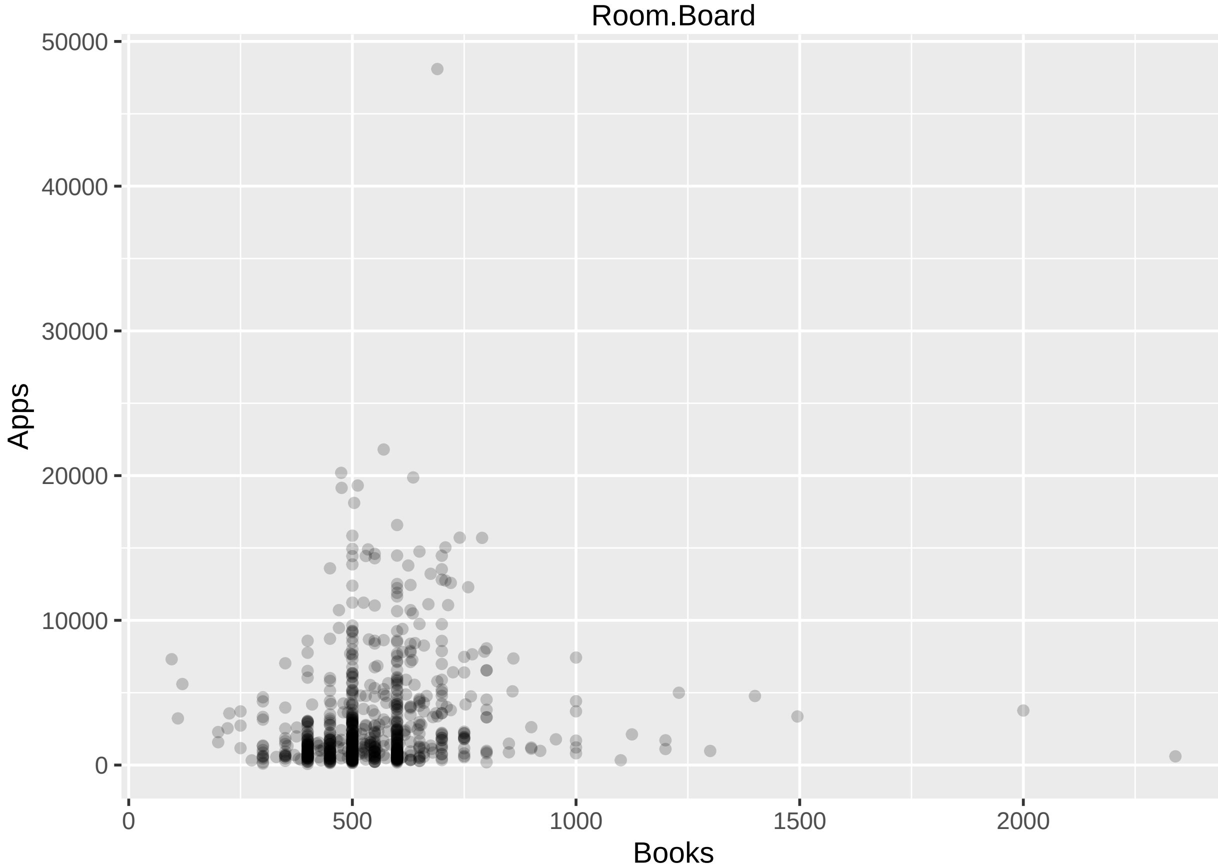

50000
40000
30000
20000
10000
0
Apps
25
50
75
100
Top25perc

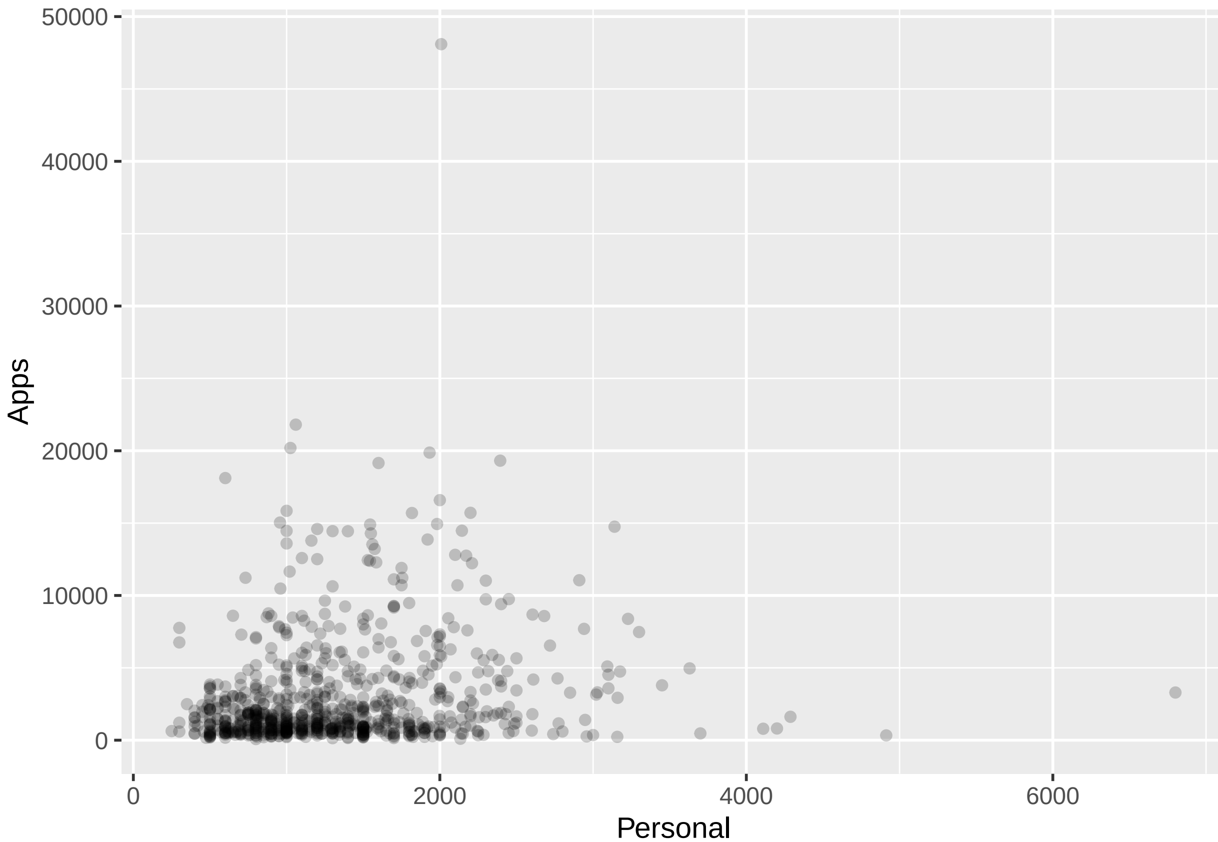

50000
40000
30000
20000
10000
0
Apps
0
10000
20000
30000
F.Undergrad

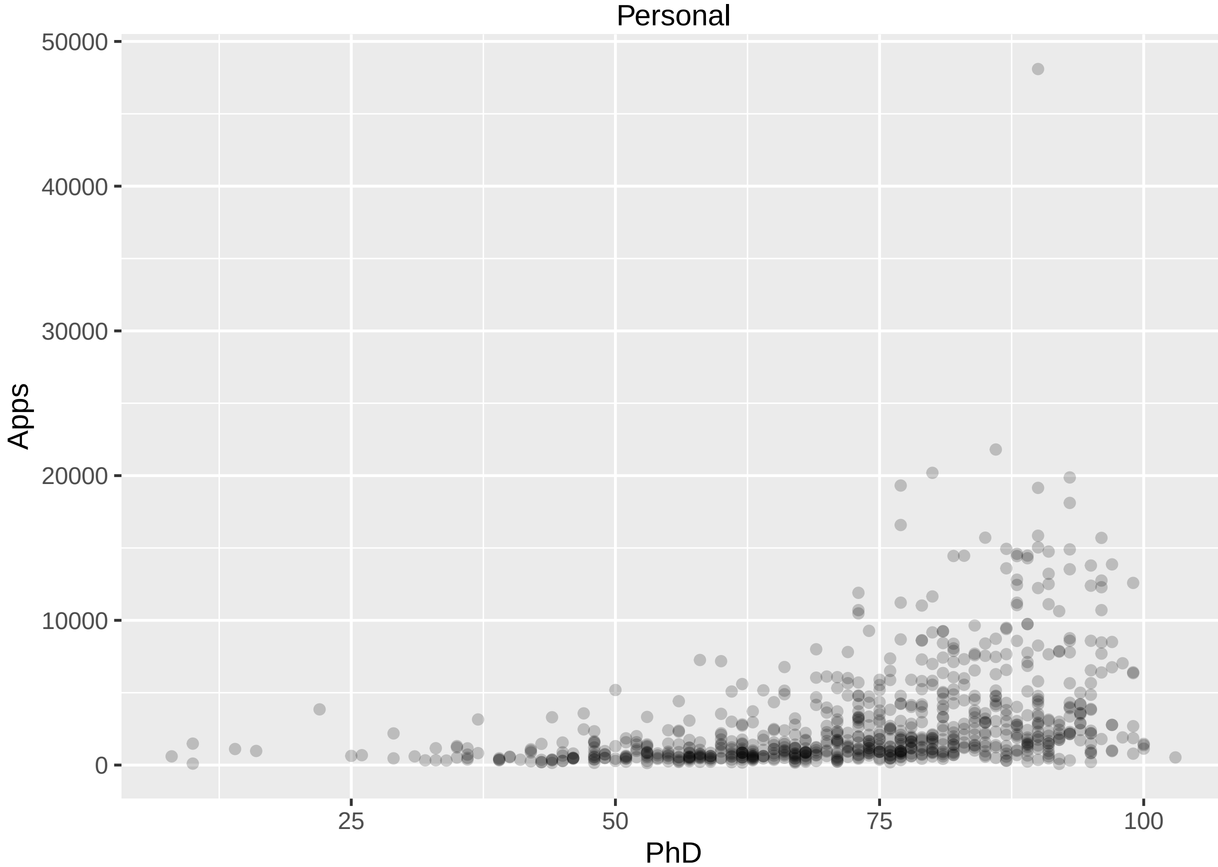

Apps
P.Undergrad
50000
40000
30000
20000
10000
0
0
5000
10000
15000
20000

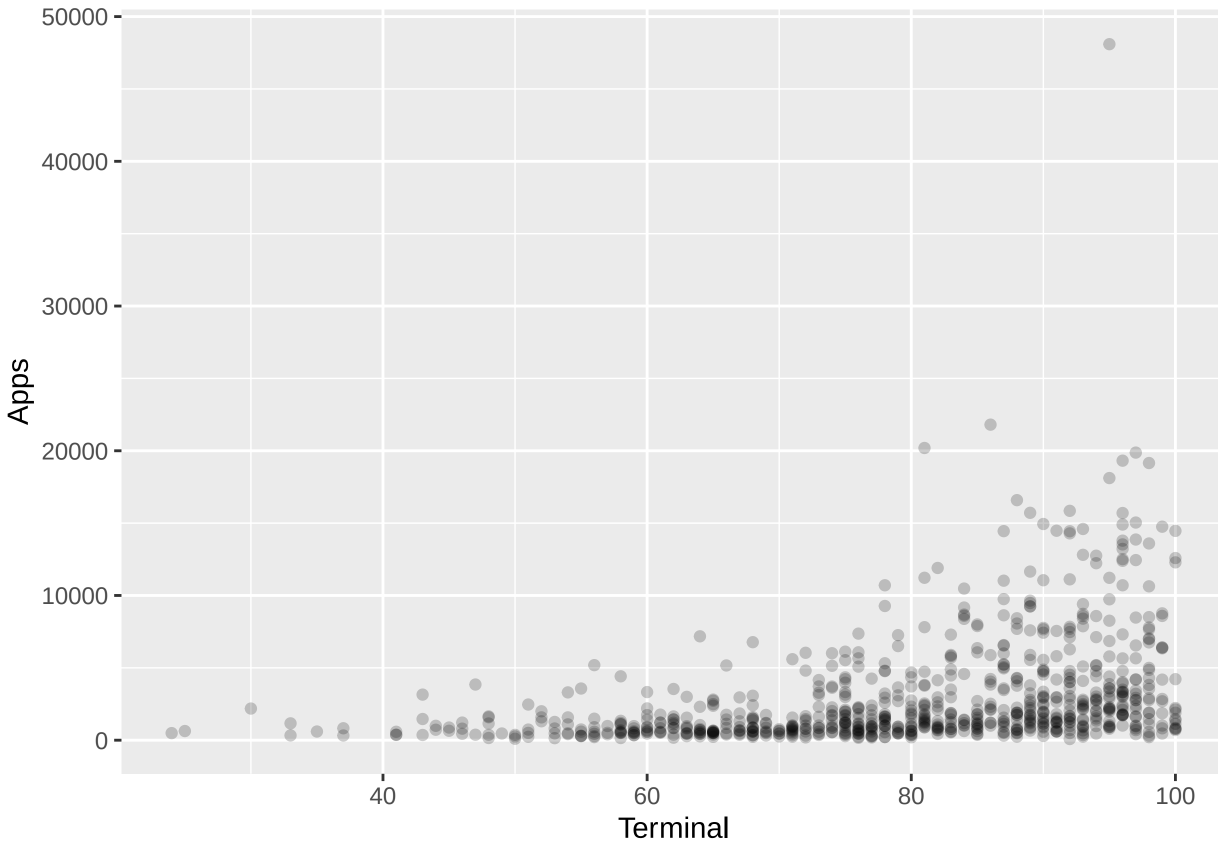

Apps
Outstate
50000
40000
30000
20000
10000
0
5000
10000
15000
20000

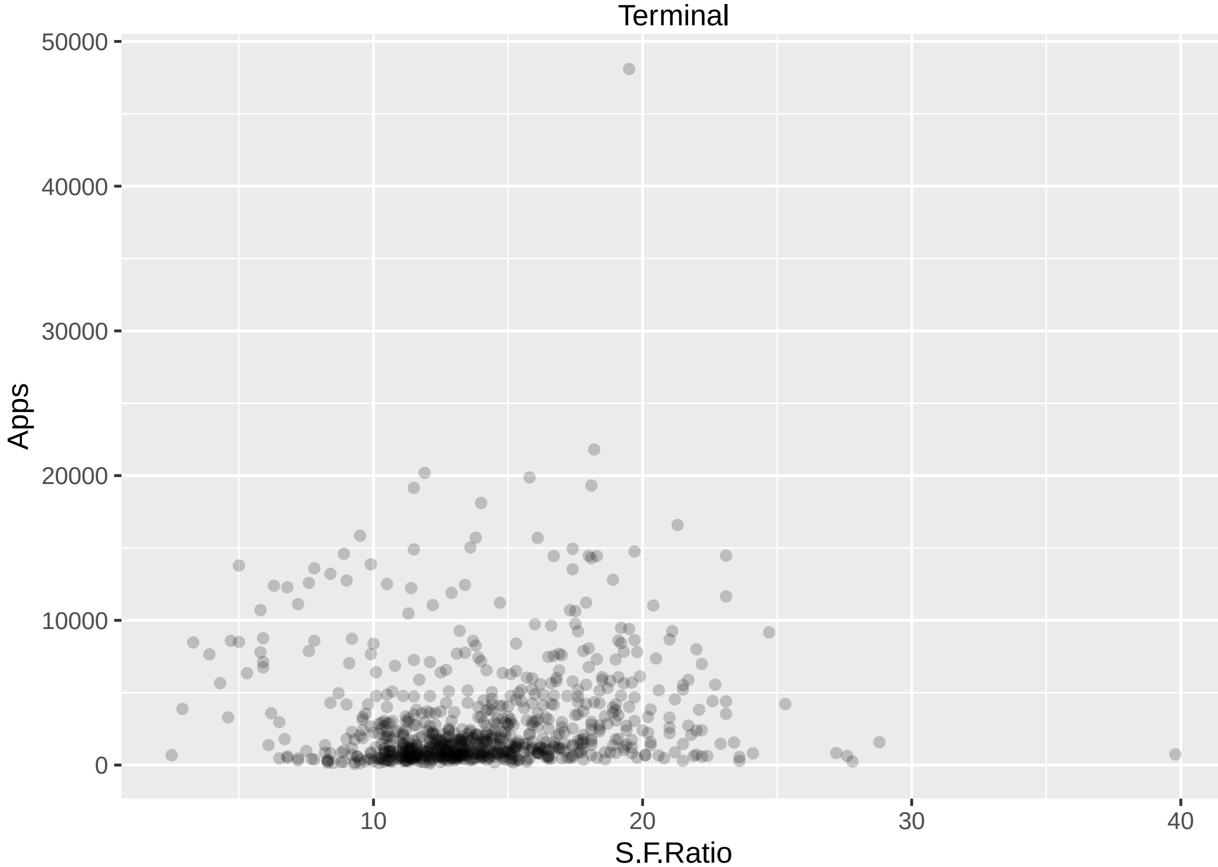

50000
40000
30000
20000
10000
0
Apps
2000
4000
6000
8000
Room.Board

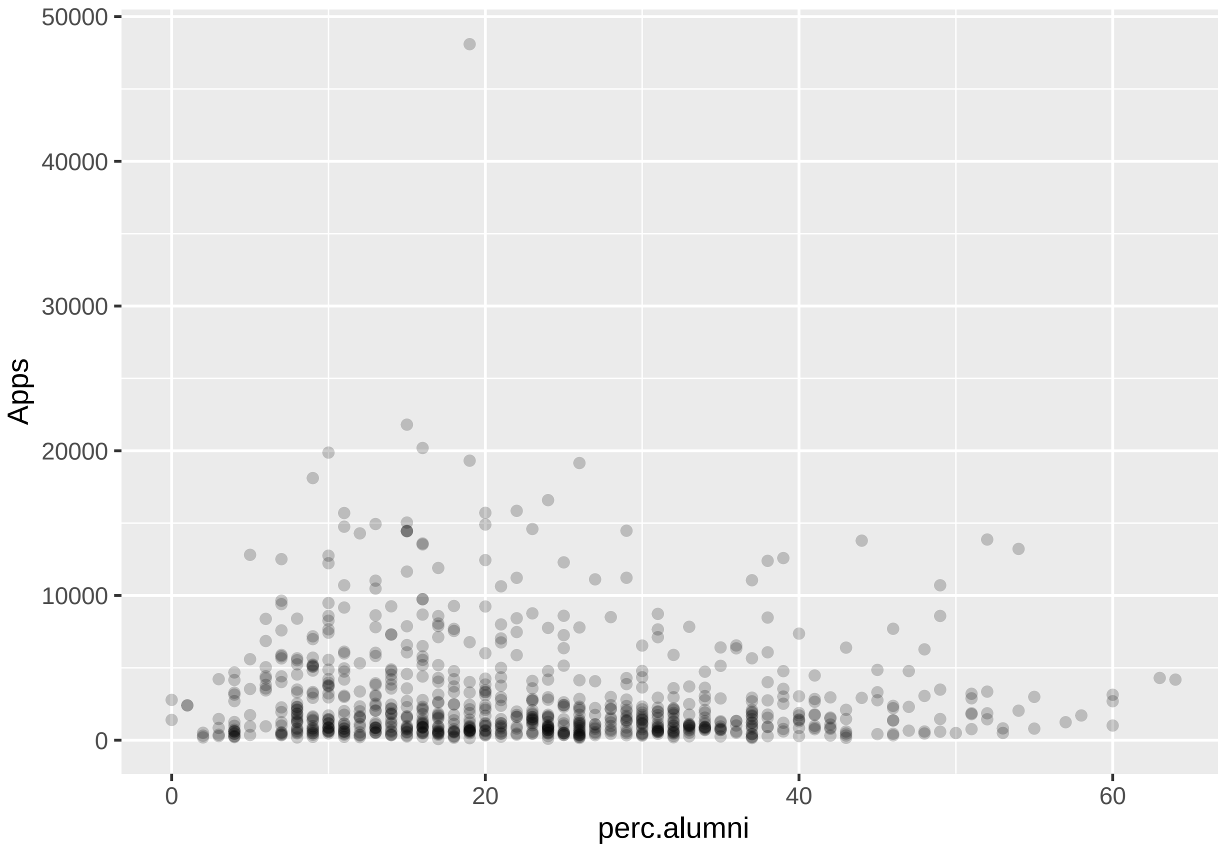

50000
40000
30000
20000
10000
0
Apps
0
500
1000
1500
2000
Books

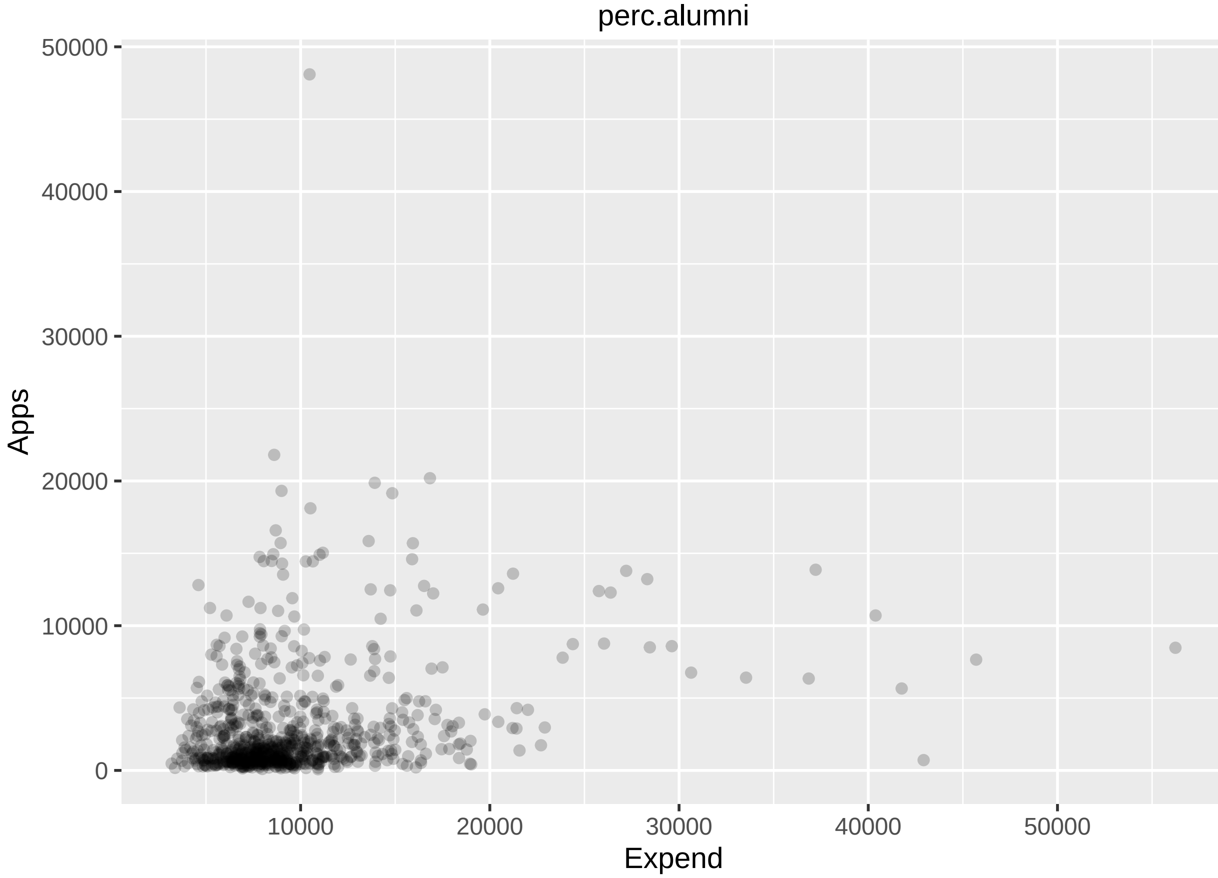

50000
40000
30000
20000
10000
0
Apps
0
2000
4000
6000
Personal

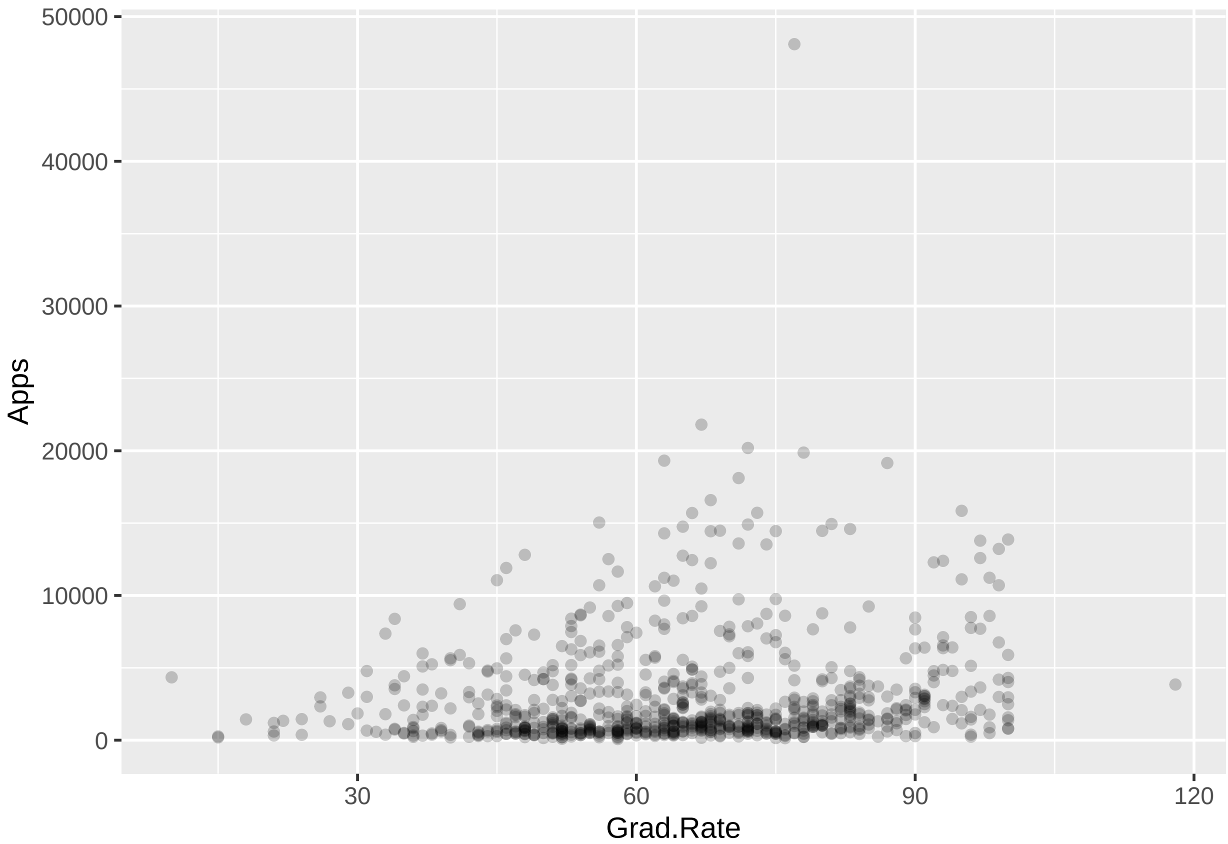

50000
40000
30000
20000
10000
0
Apps
25
50
75
100
PhD


50000
40000
30000
20000
10000
0
Apps
40
60
80
100
Terminal


50000
40000
30000
20000
10000
0
Apps
10
20
30
40
S.F.Ratio


50000
40000
30000
20000
10000
0
Apps
0
20
40
60
perc.alumni


50000
40000
30000
20000
10000
0
Apps
10000
20000
30000
40000
50000
Expend



There are some non linear looking relationships. Let's look at books.

```
# Recipe
lasso_pl_recipe <-
  recipe(formula = Apps ~ Books, data = College_train) %>%
  step_poly(all_numeric_predictors(), degree = 2, options = list(raw = TRUE)) %>%
  step_normalize(all_predictors())

# Specification
lasso_pl_spec <-
  linear_reg(mixture = 1, penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

# Workflow
lasso_pl_workflow <- workflow() %>%
  add_recipe(lasso_pl_recipe) %>%
  add_model(lasso_pl_spec)

# Grid
penalty_grid <- grid_regular(
  penalty(range = c(-5, 2)),
  levels = 50
)

# Tune model
tune_res <- tune_grid(
  lasso_pl_workflow,
```

```
  resamples = College_fold,
  grid = penalty_grid
)

# Finalize model
lasso_pl_final <- finalize_workflow(lasso_pl_workflow, best_penalty)
lasso_pl_final_fit <- fit(lasso_pl_final, data = College_train)

# Check RMSE
augment(lasso_pl_final_fit, new_data = College_test) %>%
  rmse(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       3434.
```

```
books_min <- min(College$Books)
books_max <- max(College$Books)
books_range <- tibble(Books = seq(books_min, books_max))
books_range
```

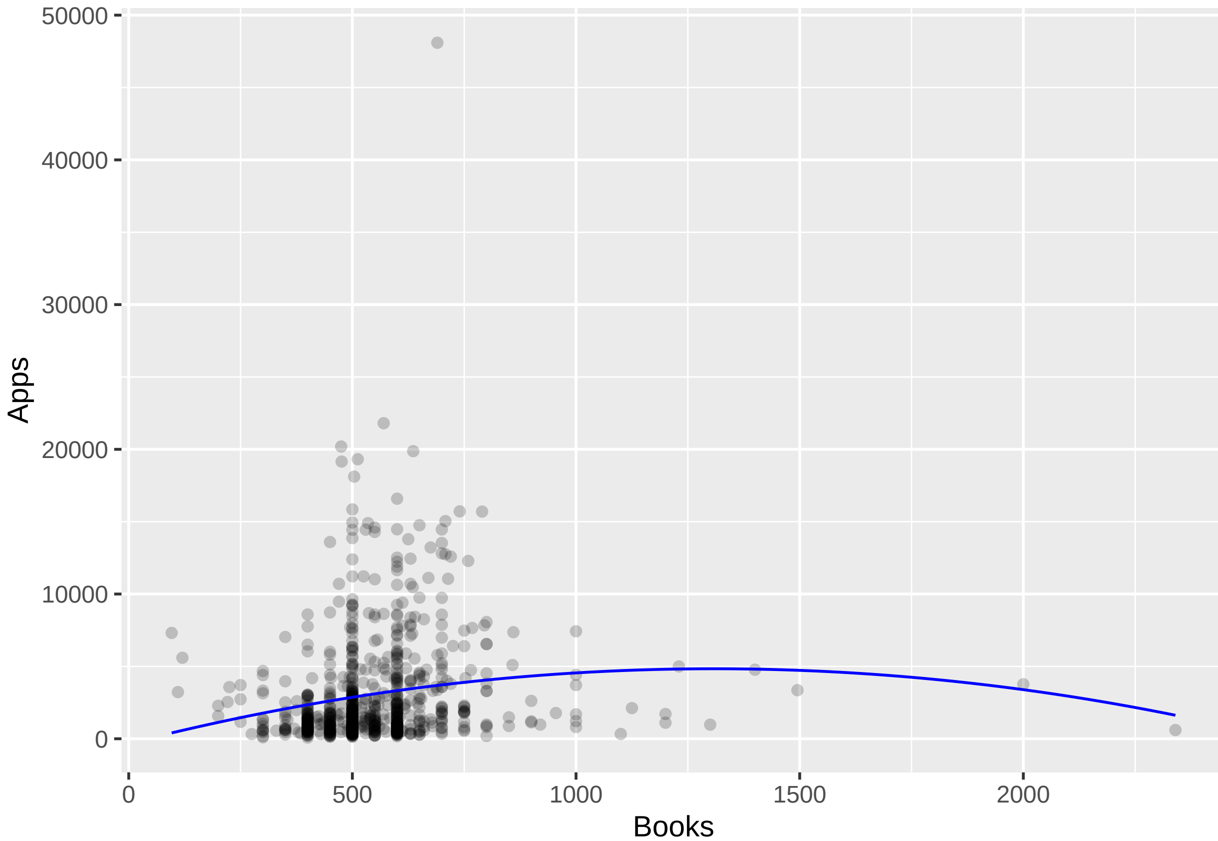
```
## # A tibble: 2,245 x 1
##    Books
##    <int>
##  1    96
##  2    97
##  3    98
##  4    99
##  5   100
##  6   101
##  7   102
##  8   103
##  9   104
## 10   105
## # ... with 2,235 more rows
```

```
regression_lines <- bind_cols(
  predict(lasso_pl_final_fit, new_data = books_range),
  books_range
)

College
```

```
## # A tibble: 777 x 18
##    Private  Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad
##    <fct>   <dbl>  <dbl>  <dbl>     <dbl>     <dbl>       <dbl>       <dbl>
##  1 Yes      1660   1232    721        23        52        2885         537
##  2 Yes      2186   1924    512        16        29        2683        1227
##  3 Yes      1428   1097    336        22        50        1036          99
##  4 Yes       417    349    137        60        89         510          63
##  5 Yes       193    146     55        16        44         249         869
##  6 Yes       587    479    158        38        62         678          41
##  7 Yes       353    340    103        17        45         416         230
##  8 Yes      1899   1720    489        37        68        1594          32
```

```
##   9 Yes       1038     839     227         30         63          973           306
##  10 Yes        582     498     172         21         44          799            78
## # ... with 767 more rows, and 10 more variables: Outstate <dbl>,
## #   Room.Board <dbl>, Books <dbl>, Personal <dbl>, PhD <dbl>, Terminal <dbl>,
## #   S.F.Ratio <dbl>, perc.alumni <dbl>, Expend <dbl>, Grad.Rate <dbl>
```

```
College %>%
  ggplot(aes(Books, Apps)) +
  geom_point(alpha = 0.2) +
  geom_line(aes(y = .pred), data = regression_lines, color = "blue")
```



As we can see a polynomial fits this data better than a linear relationship.

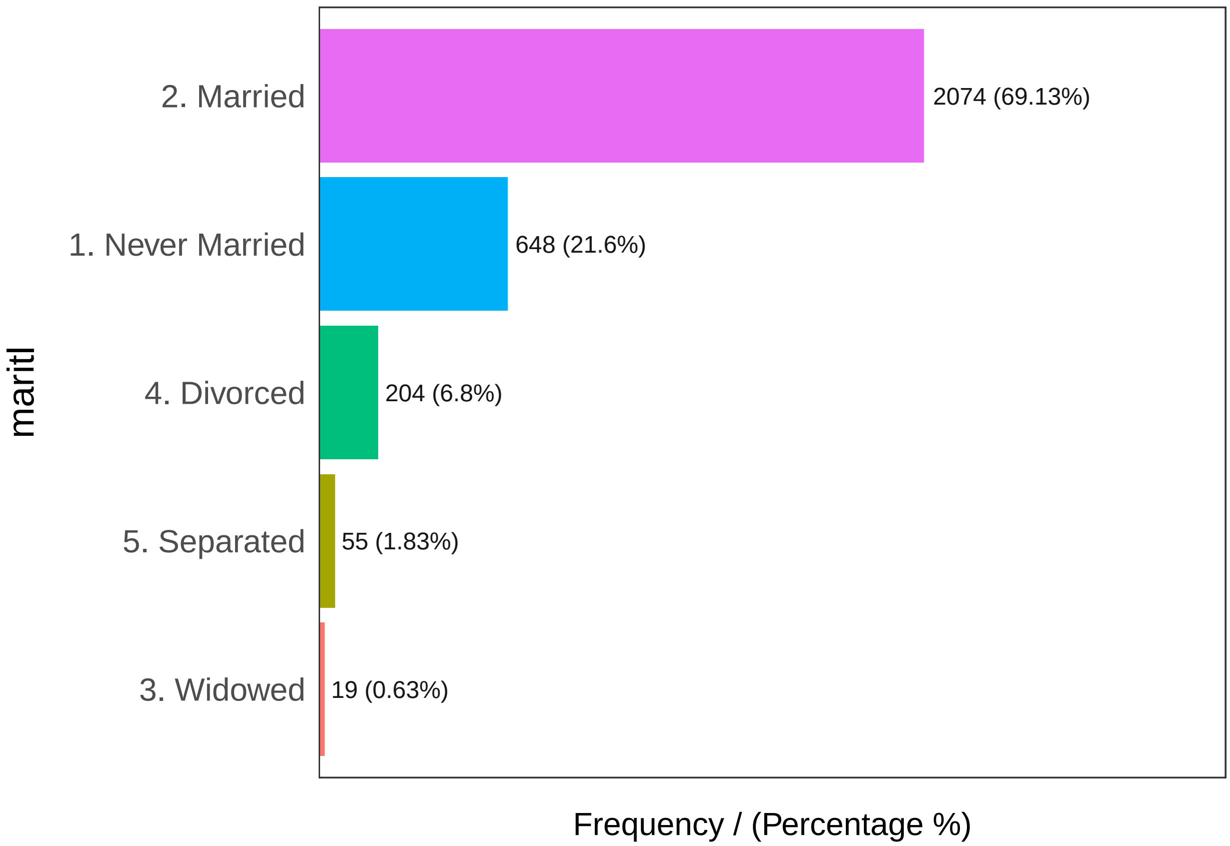
Exercise 3

The Wage data set contains a number of features, such as marital status (marital), job class (jobclass), and others. Explore the relationships between some of these predictors and wage, and use non-linear fitting techniques in order to fit flexible models to the data. Create plots of the results obtained, and write a summary of your findings.

```
Wage <- tibble(Wage)

basic_eda(Wage)
```

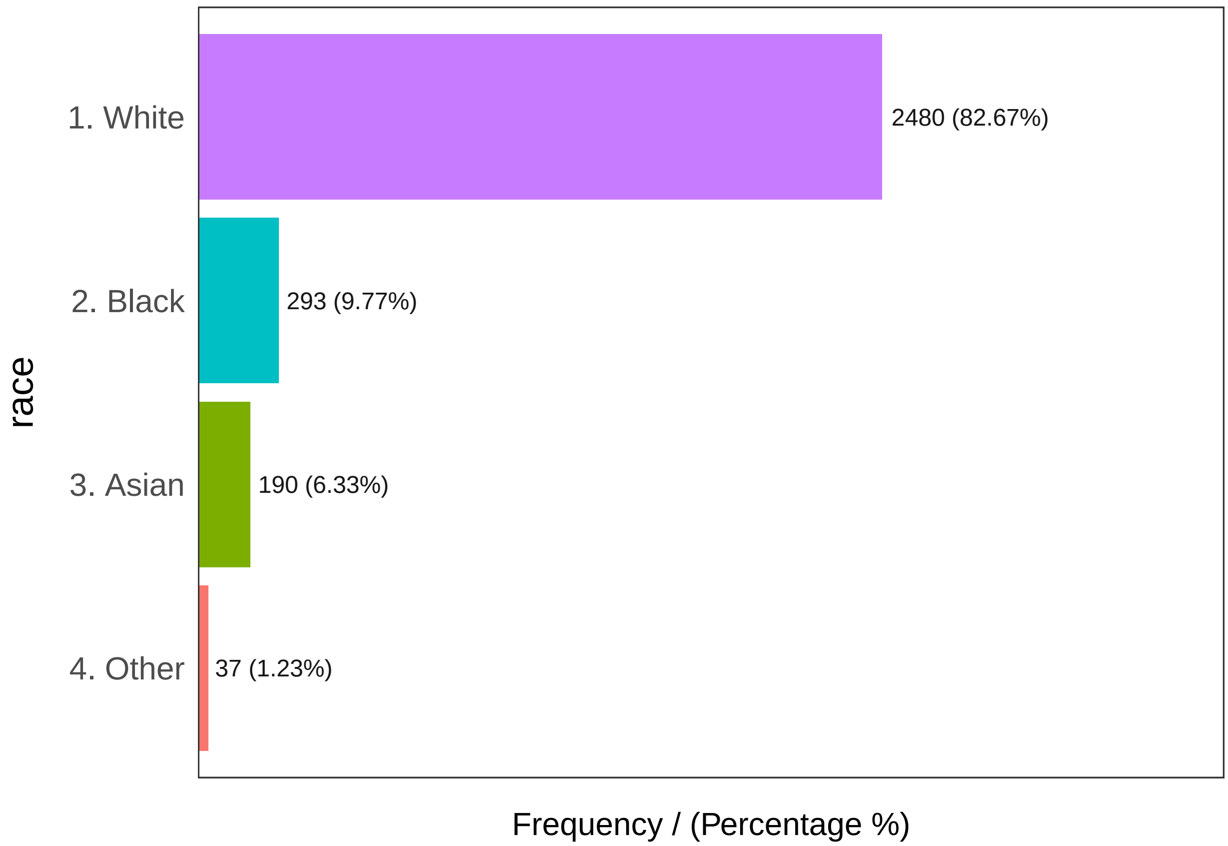
```
## Rows: 3,000
## Columns: 11
## $ year       <int> 2006, 2004, 2003, 2003, 2005, 2008, 2009, 2008, 2006, 2004,~
## $ age        <int> 18, 24, 45, 43, 50, 54, 44, 30, 41, 52, 45, 34, 35, 39, 54,~
```

```
## $ maritl     <fct> 1. Never Married, 1. Never Married, 2. Married, 2. Married,~
## $ race       <fct> 1. White, 1. White, 1. White, 3. Asian, 1. White, 1. White,~
## $ education  <fct> 1. < HS Grad, 4. College Grad, 3. Some College, 4. College ~
## $ region     <fct> 2. Middle Atlantic, 2. Middle Atlantic, 2. Middle Atlantic,~
## $ jobclass   <fct> 1. Industrial, 2. Information, 1. Industrial, 2. Informatio~
## $ health     <fct> 1. <=Good, 2. >=Very Good, 1. <=Good, 2. >=Very Good, 1. <=~
## $ health_ins <fct> 2. No, 2. No, 1. Yes, 1. Yes, 1. Yes, 1. Yes, 1. Yes, 1. Ye~
## $ logwage    <dbl> 4.318063, 4.255273, 4.875061, 5.041393, 4.318063, 4.845098,~
## $ wage       <dbl> 75.04315, 70.47602, 130.98218, 154.68529, 75.04315, 127.115~
##              variable q_zeros p_zeros q_na p_na q_inf p_inf    type unique
## year             year       0       0    0    0     0     0 integer      7
## age               age       0       0    0    0     0     0 integer     61
## maritl         maritl       0       0    0    0     0     0  factor      5
## race             race       0       0    0    0     0     0  factor      4
## education   education       0       0    0    0     0     0  factor      5
## region         region       0       0    0    0     0     0  factor      1
## jobclass     jobclass       0       0    0    0     0     0  factor      2
## health         health       0       0    0    0     0     0  factor      2
## health_ins health_ins       0       0    0    0     0     0  factor      2
## logwage       logwage       0       0    0    0     0     0 numeric    508
## wage             wage       0       0    0    0     0     0 numeric    508

## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.

##             maritl frequency percentage cumulative_perc
## 1       2. Married      2074      69.13           69.13
## 2 1. Never Married       648      21.60           90.73
## 3      4. Divorced       204       6.80           97.53
## 4     5. Separated        55       1.83           99.36
## 5       3. Widowed        19       0.63          100.00

## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



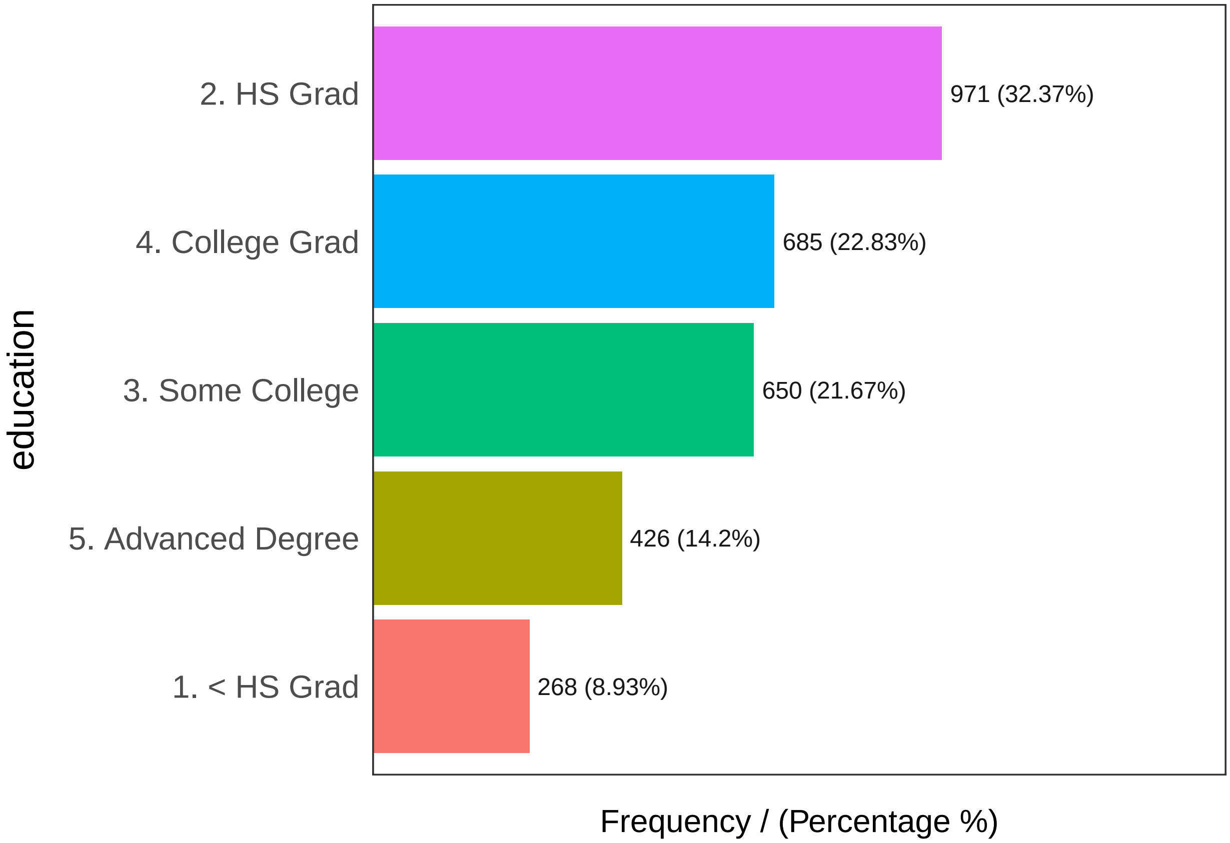

```
##        race frequency percentage cumulative_perc
## 1 1. White      2480      82.67           82.67
## 2 2. Black       293       9.77           92.44
## 3 3. Asian       190       6.33           98.77
## 4 4. Other        37       1.23          100.00
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



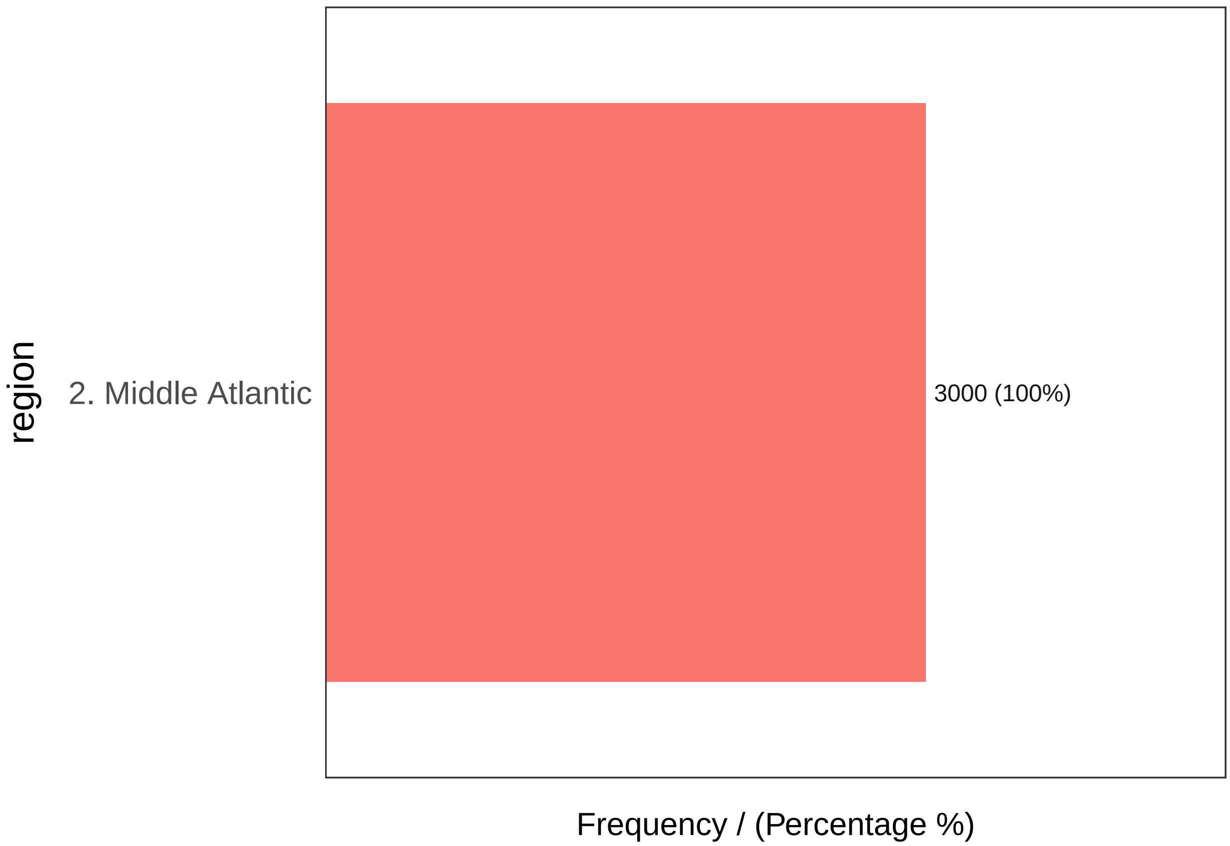

```
##             education frequency percentage cumulative_perc
## 1          2. HS Grad       971      32.37           32.37
## 2     4. College Grad       685      22.83           55.20
## 3     3. Some College       650      21.67           76.87
## 4 5. Advanced Degree       426      14.20           91.07
## 5        1. < HS Grad       268       8.93          100.00
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



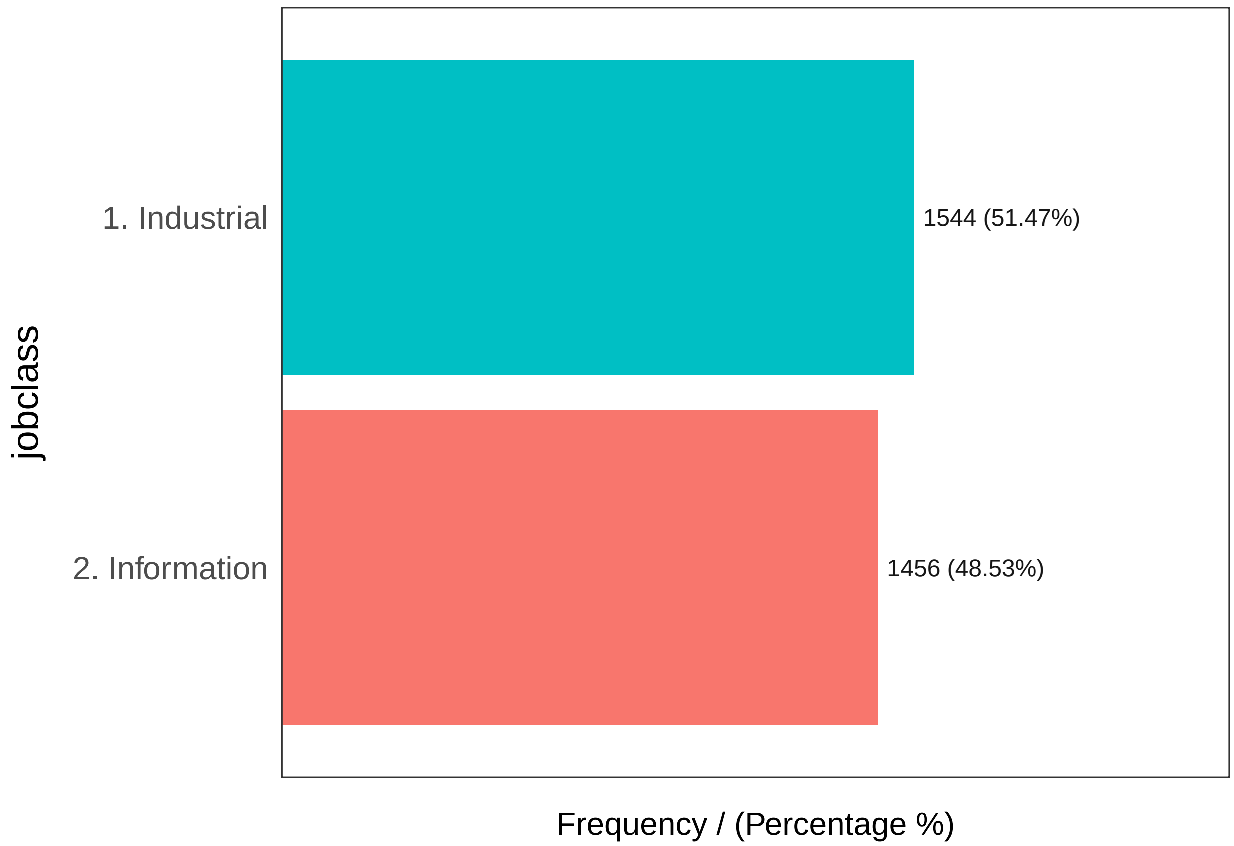

```
##                region frequency percentage cumulative_perc
## 1 2. Middle Atlantic      3000        100             100
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



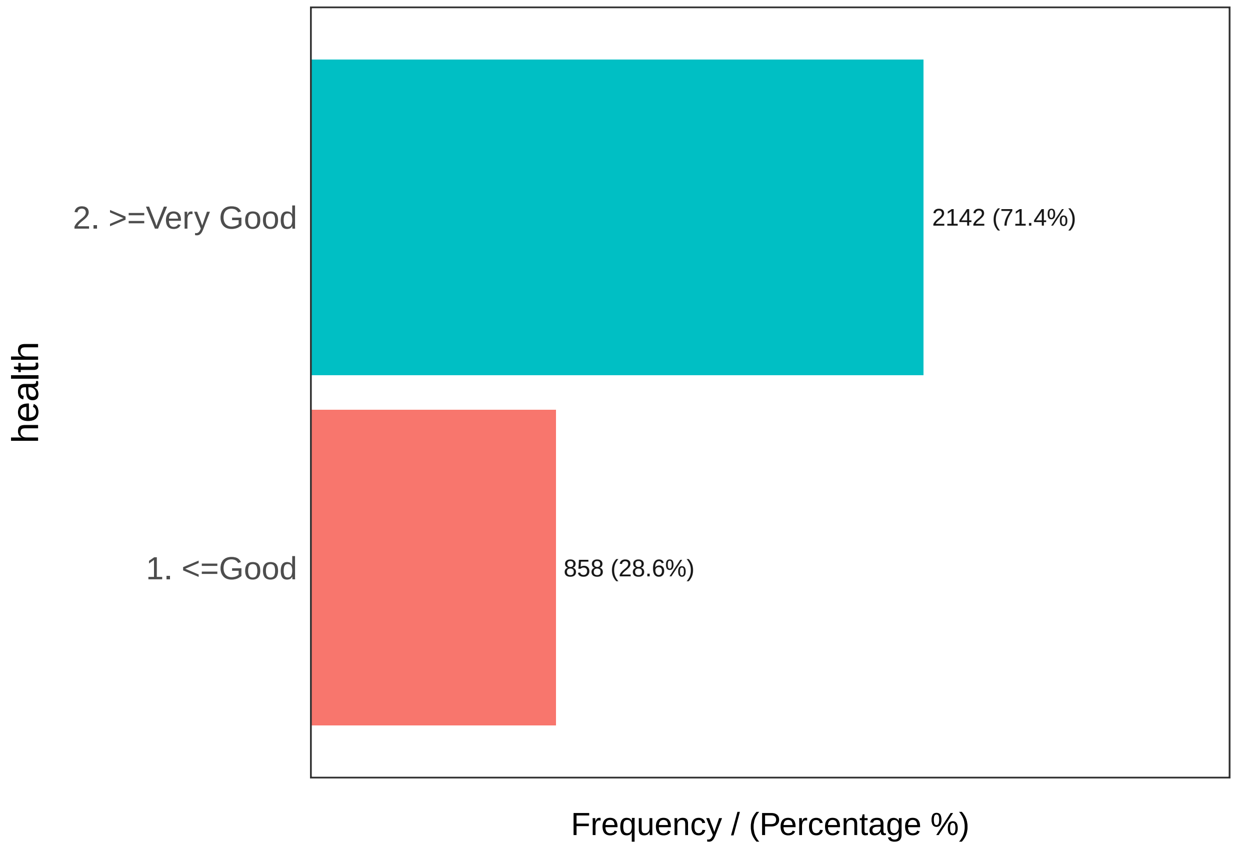

```
##         jobclass frequency percentage cumulative_perc
## 1  1. Industrial      1544      51.47           51.47
## 2 2. Information      1456      48.53          100.00
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



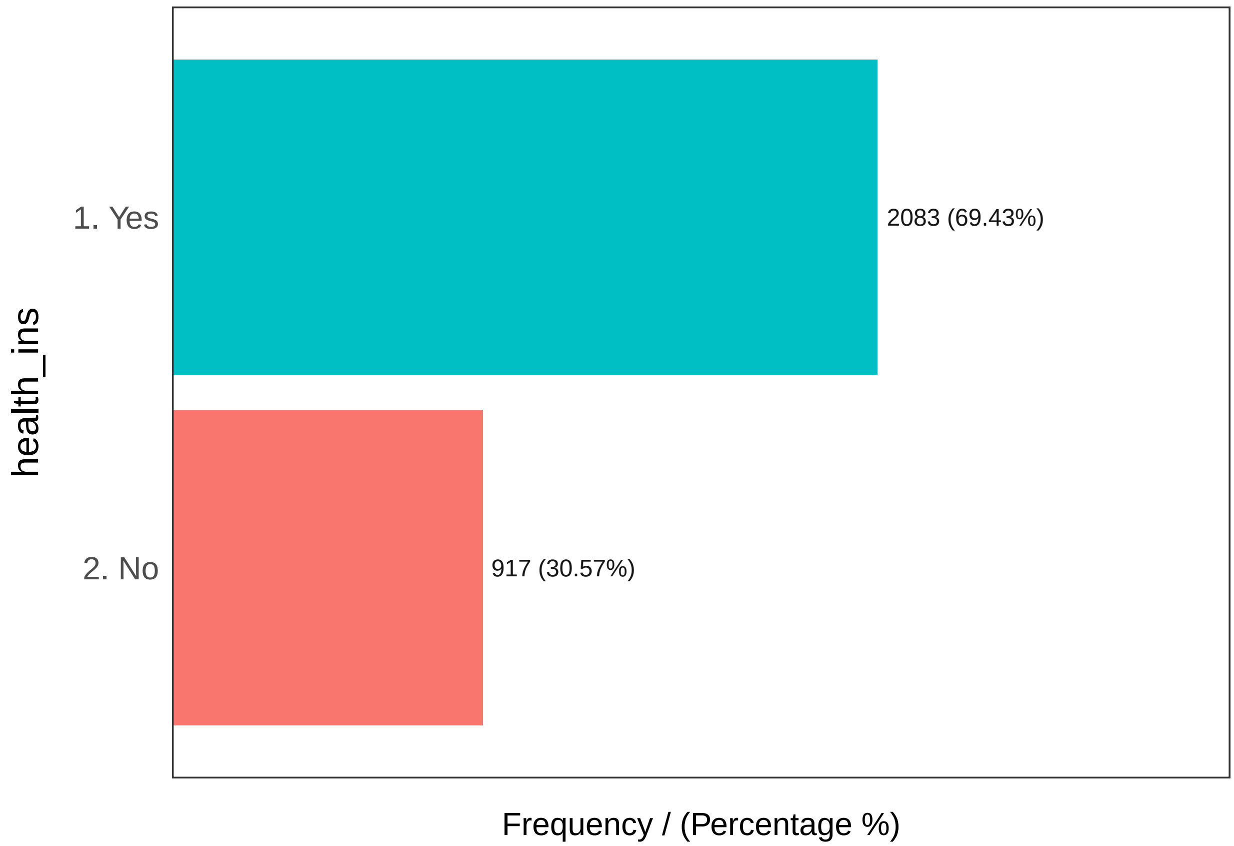

```
##           health frequency percentage cumulative_perc
## 1 2. >=Very Good      2142       71.4            71.4
## 2      1. <=Good       858       28.6           100.0
```

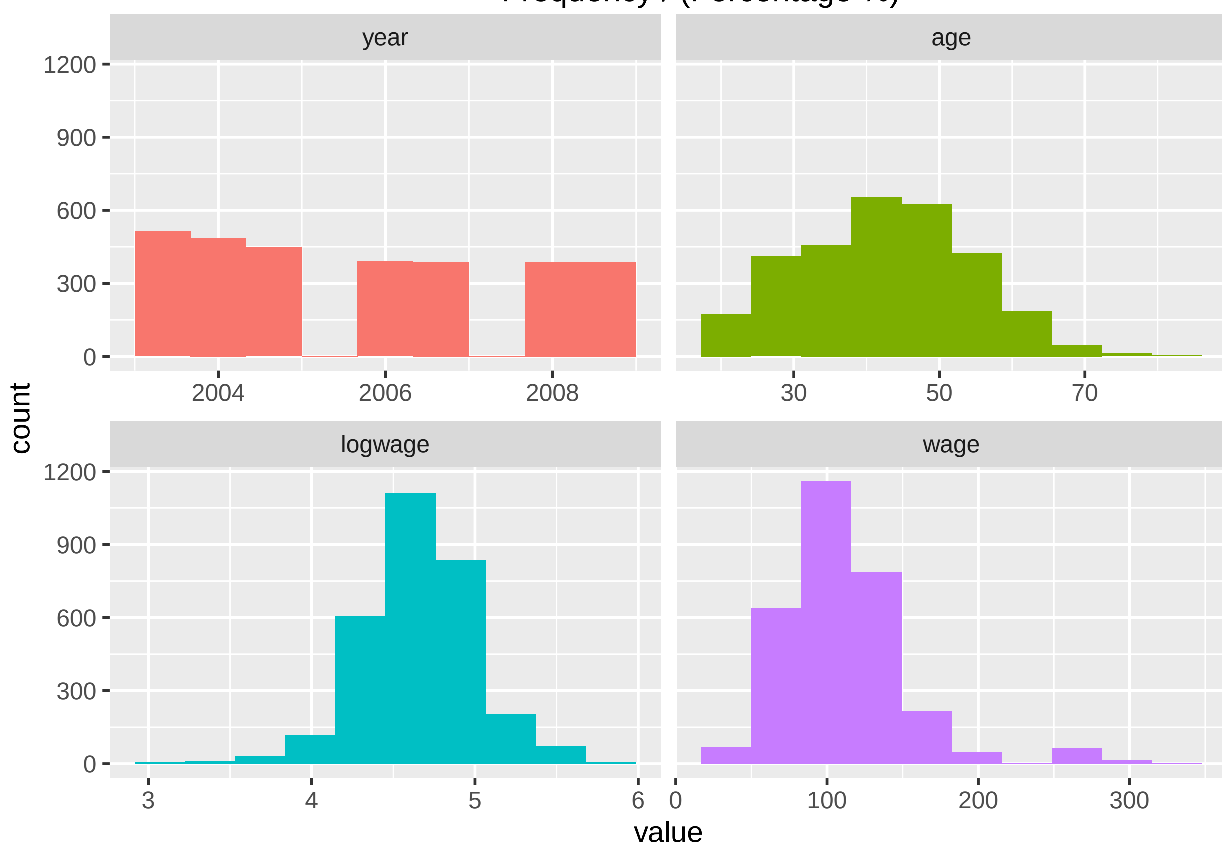
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```




```
##   health_ins frequency percentage cumulative_perc
## 1     1. Yes      2083      69.43           69.43
## 2      2. No       917      30.57          100.00
##
##   variable        mean    std_dev variation_coef        p_01        p_05
## 1     year 2005.791000  2.0261673    0.001010159 2003.000000 2003.000000
## 2      age   42.414667 11.5424056    0.272132414   20.000000   24.000000
## 3  logwage    4.653905  0.3517526    0.075582244    3.698918    4.113943
## 4     wage  111.703608 41.7285955    0.373565332   40.403552   61.187526
##          p_25        p_50        p_75        p_95        p_99   skewness
## 1 2004.000000 2006.000000 2008.000000 2009.000000 2009.000000  0.1428111
## 2   33.750000   42.000000   51.000000   61.000000   70.000000  0.1477340
## 3    4.447158    4.653213    4.857332    5.176091    5.626186 -0.1235535
## 4   85.383940  104.921507  128.680488  176.989650  277.601418  1.6814889
##   kurtosis        iqr                            range_98
## 1 1.733853  4.0000000                        [2003, 2009]
## 2 2.552129 17.2500000                            [20, 70]
## 3 4.728038  0.4101745  [3.6989175737819, 5.62618633492728]
## 4 7.828952 43.2965478 [40.4035523122557, 277.601417511009]
##                                  range_80
## 1                            [2003, 2009]
## 2                                [27, 58]
## 3 [4.25527250510331, 5.04151096788643]
## 4 [70.4760196469445, 154.703600419223]
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```


health_ins
1. Yes
2. No
2083 (69.43%)
917 (30.57%)
Frequency / (Percentage %)


year
age
logwage
wage
count
value
1200
900
600
300
0
2004
2006
2008
30
50
70
3
4
5
6
0
100
200
300

```
## data
```

```
##
##  11  Variables      3000  Observations
## ---------------------------------------------------------------------------------
## year
##        n  missing distinct     Info     Mean      Gmd
##     3000        0        7    0.979     2006    2.312
##
## lowest : 2003 2004 2005 2006 2007, highest: 2005 2006 2007 2008 2009
##
## Value       2003  2004  2005  2006  2007  2008  2009
## Frequency    513   485   447   392   386   388   389
## Proportion 0.171 0.162 0.149 0.131 0.129 0.129 0.130
## ---------------------------------------------------------------------------------
## age
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3000        0       61    0.999    42.41    13.16    24.00    27.00
##      .25      .50      .75      .90      .95
##    33.75    42.00    51.00    58.00    61.00
##
## lowest : 18 19 20 21 22, highest: 74 75 76 77 80
## ---------------------------------------------------------------------------------
## maritl
##        n  missing distinct
##     3000        0        5
##
## lowest : 1. Never Married 2. Married       3. Widowed       4. Divorced      5. Separated
## highest: 1. Never Married 2. Married       3. Widowed       4. Divorced      5. Separated
##
## Value      1. Never Married       2. Married       3. Widowed      4. Divorced
## Frequency               648             2074               19              204
## Proportion            0.216            0.691            0.006            0.068
##
## Value          5. Separated
## Frequency                55
## Proportion            0.018
## ---------------------------------------------------------------------------------
## race
##        n  missing distinct
##     3000        0        4
##
## Value      1. White 2. Black 3. Asian 4. Other
## Frequency      2480      293      190       37
## Proportion    0.827    0.098    0.063    0.012
## ---------------------------------------------------------------------------------
## education
##        n  missing distinct
##     3000        0        5
##
## lowest : 1. < HS Grad       2. HS Grad         3. Some College    4. College Grad    5. Advanced Deg
## highest: 1. < HS Grad       2. HS Grad         3. Some College    4. College Grad    5. Advanced Deg
##
## Value             1. < HS Grad         2. HS Grad    3. Some College
## Frequency                  268                971                650
## Proportion               0.089              0.324              0.217
```

```
##
## Value         4. College Grad 5. Advanced Degree
## Frequency                 685                426
## Proportion              0.228              0.142
## ---------------------------------------------------------------------------------
## region
##                 n           missing          distinct              value
##              3000                 0                 1 2. Middle Atlantic
##
## Value      2. Middle Atlantic
## Frequency                3000
## Proportion                  1
## ---------------------------------------------------------------------------------
## jobclass
##        n  missing distinct
##     3000        0        2
##
## Value       1. Industrial 2. Information
## Frequency            1544           1456
## Proportion          0.515          0.485
## ---------------------------------------------------------------------------------
## health
##        n  missing distinct
##     3000        0        2
##
## Value           1. <=Good 2. >=Very Good
## Frequency             858           2142
## Proportion          0.286          0.714
## ---------------------------------------------------------------------------------
## health_ins
##        n  missing distinct
##     3000        0        2
##
## Value      1. Yes  2. No
## Frequency    2083    917
## Proportion  0.694  0.306
## ---------------------------------------------------------------------------------
## logwage
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3000        0      508        1    4.654   0.3824    4.114    4.255
##      .25      .50      .75      .90      .95
##    4.447    4.653    4.857    5.042    5.176
##
## lowest : 3.000000 3.041393 3.133858 3.147367 3.176091
## highest: 5.701323 5.735190 5.742793 5.750441 5.763128
## ---------------------------------------------------------------------------------
## wage
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     3000        0      508        1    111.7    42.64    61.19    70.48
##      .25      .50      .75      .90      .95
##    85.38   104.92   128.68   154.70   176.99
##
## lowest :  20.08554  20.93438  22.96240  23.27470  23.95294
## highest: 299.26298 309.57177 311.93457 314.32934 318.34243
```

```
## --------------------------------------------------------------------------------
```

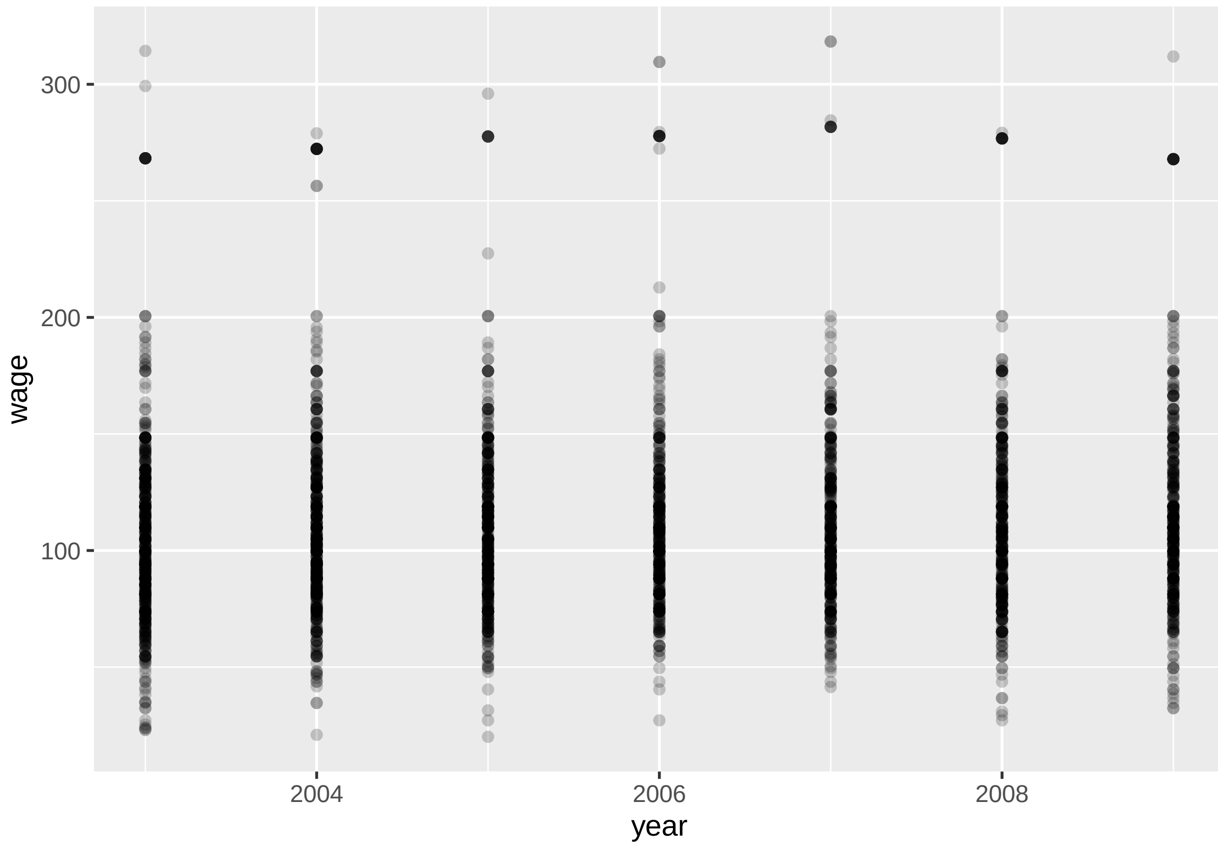
```
Wage_split <- initial_split(Wage, strata = wage, prop = 0.5)
Wage_split
```

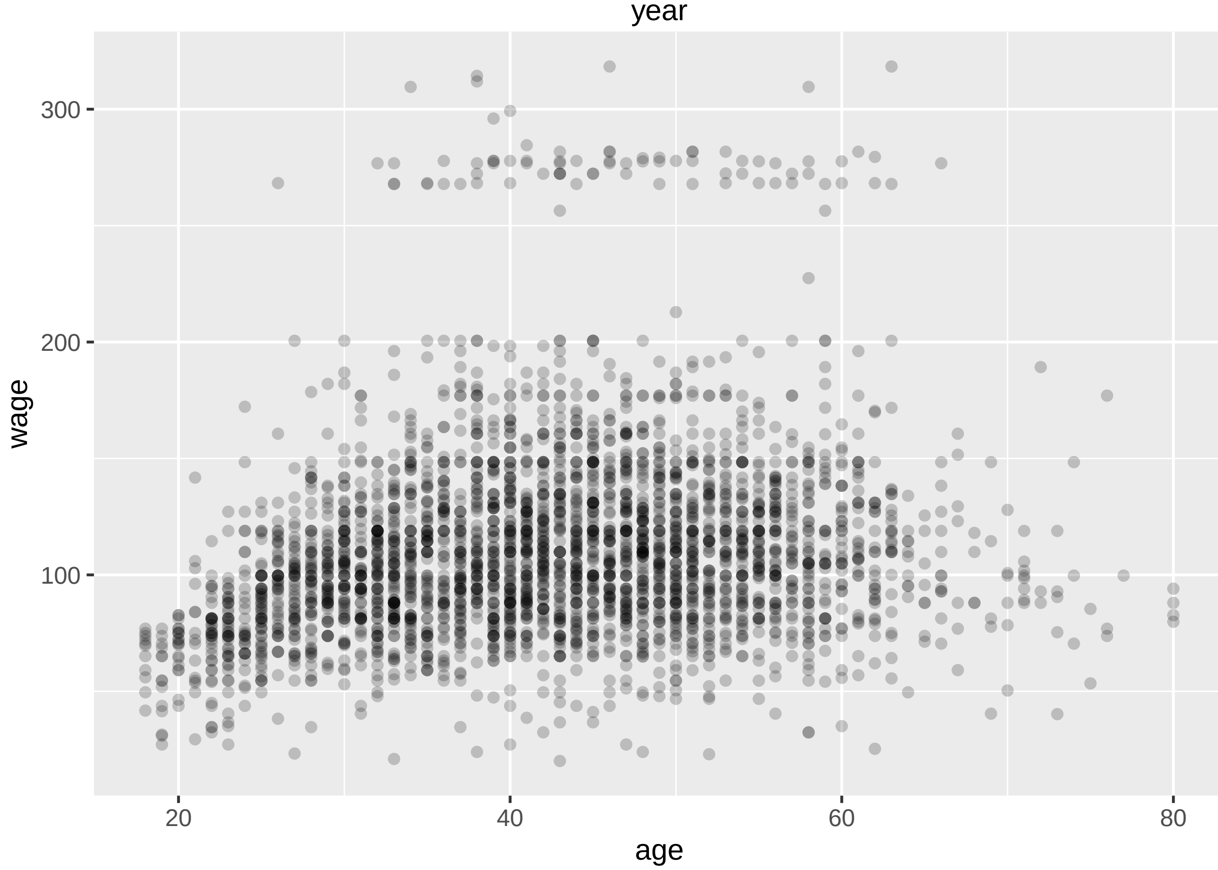
```
## <Analysis/Assess/Total>
## <1499/1501/3000>
```

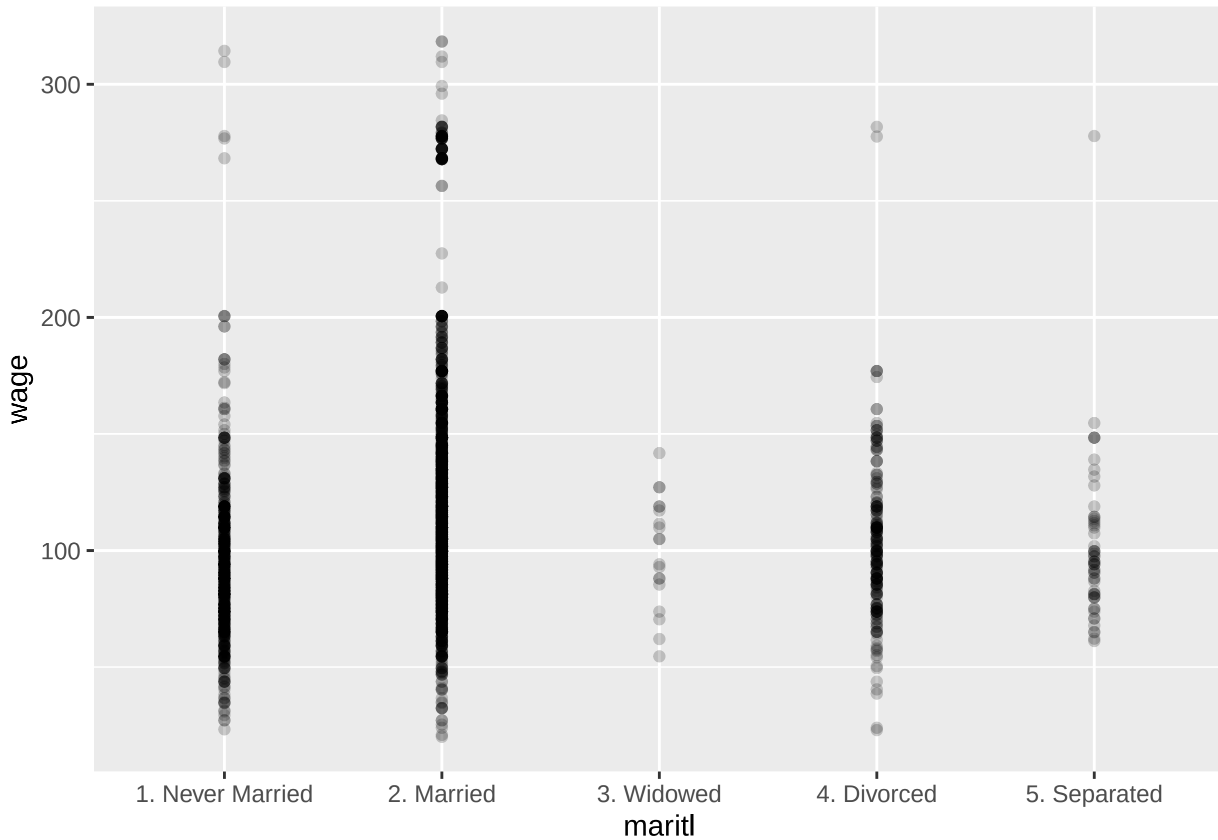
```
Wage_train <- training(Wage_split)
Wage_test <- testing(Wage_split)

Wage_fold <- vfold_cv(Wage_train, v = 10)
Wage_fold
```

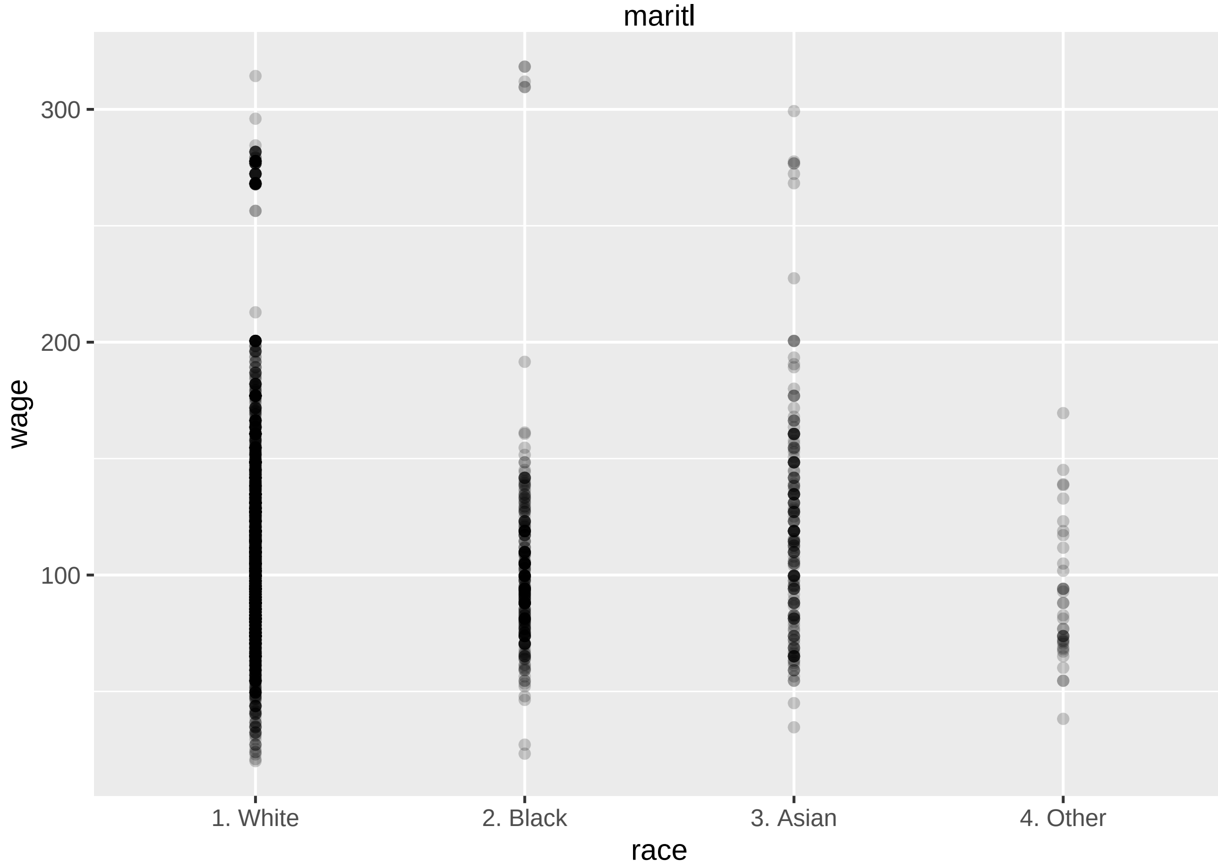
```
## #  10-fold cross-validation
## # A tibble: 10 x 2
##    splits             id
##    <list>             <chr>
##  1 <split [1349/150]> Fold01
##  2 <split [1349/150]> Fold02
##  3 <split [1349/150]> Fold03
##  4 <split [1349/150]> Fold04
##  5 <split [1349/150]> Fold05
##  6 <split [1349/150]> Fold06
##  7 <split [1349/150]> Fold07
##  8 <split [1349/150]> Fold08
##  9 <split [1349/150]> Fold09
## 10 <split [1350/149]> Fold10
```

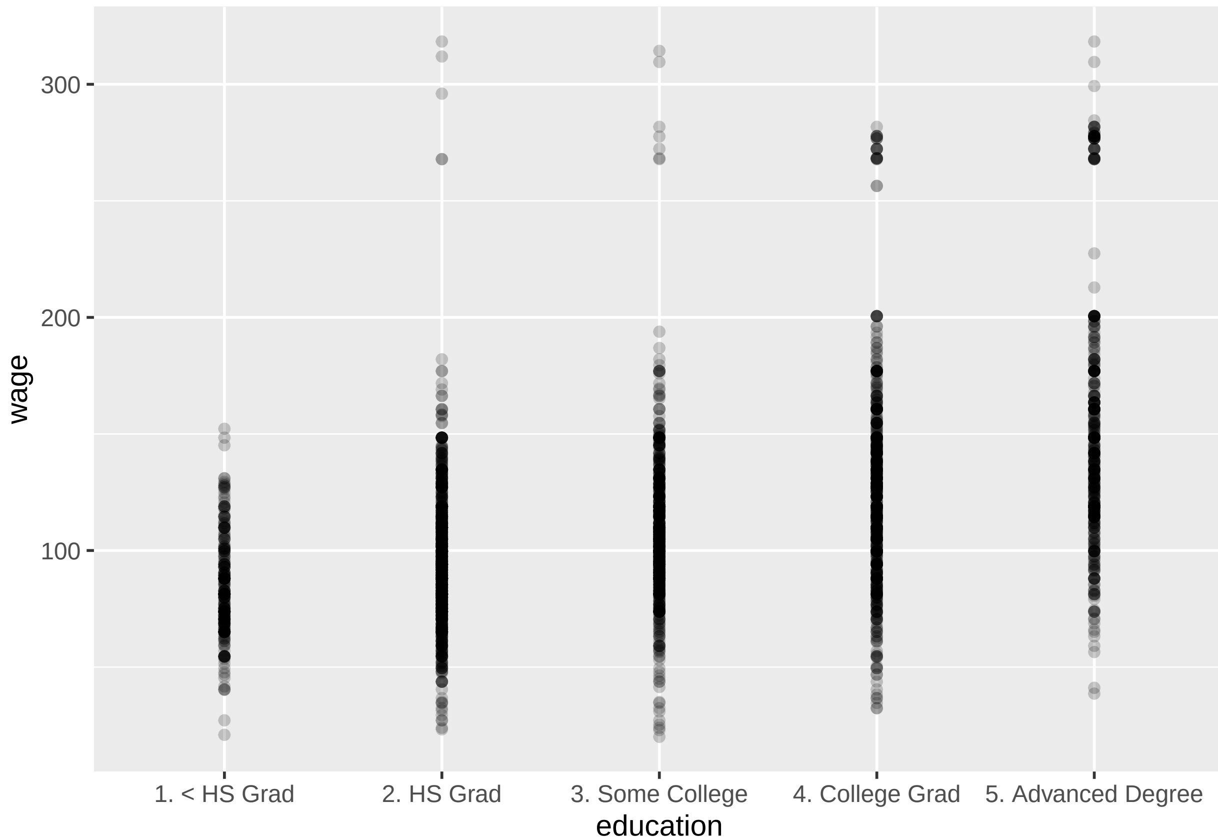
```
colnames <- colnames(Wage)
colnames <- colnames[colnames != "wage"]

for (colname in colnames) {
  pl <- Wage %>%
    ggplot(aes_string(x = colname, y = "wage")) +
    geom_point(alpha = 0.2)
  print(pl)
}
```

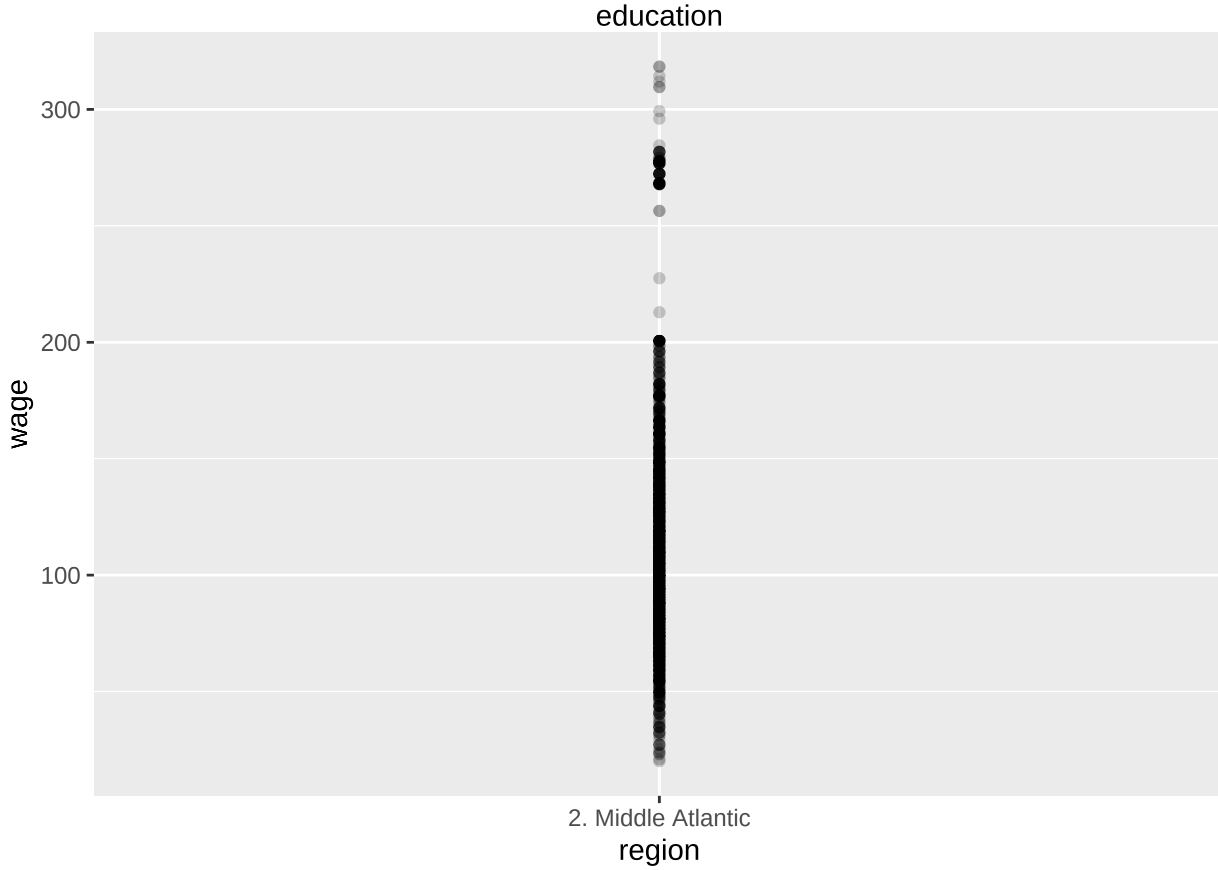

300
200
100
wage
2004
2006
2008
year

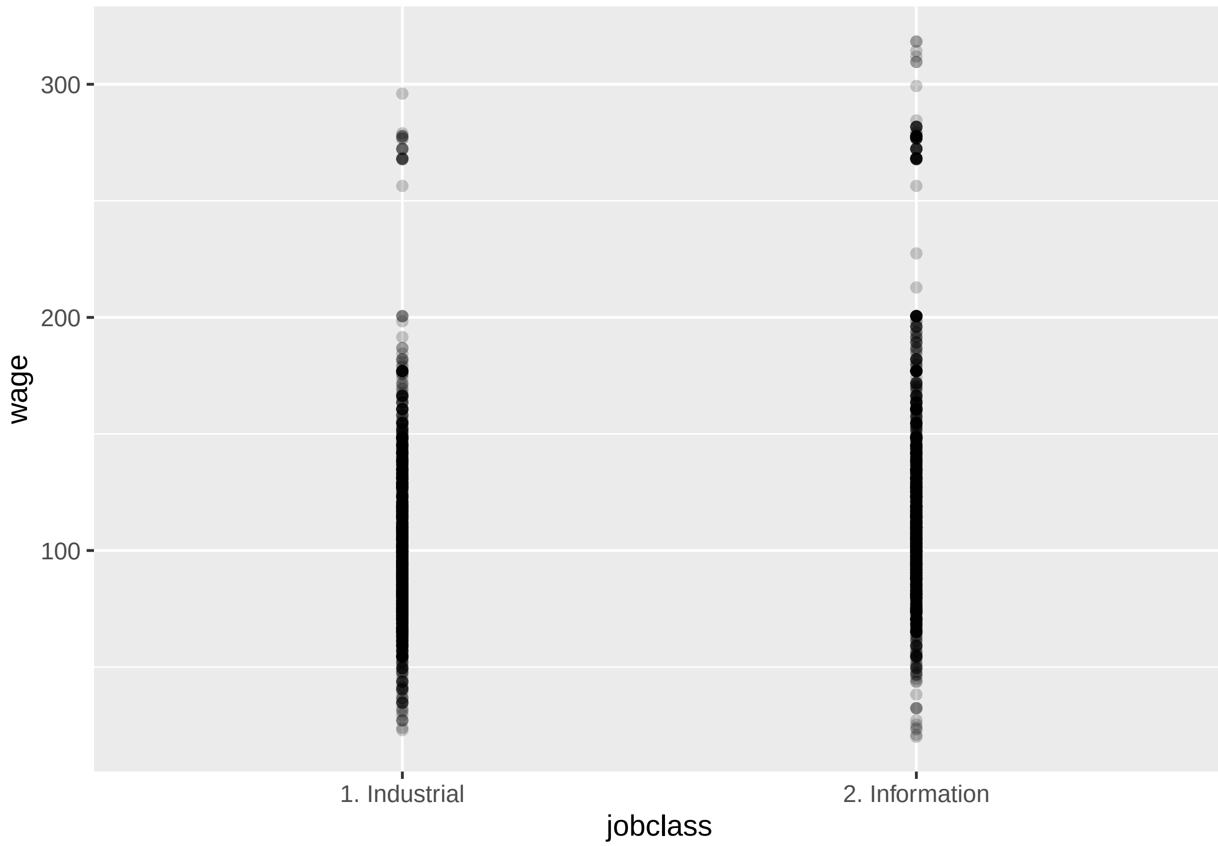

300
200
100
wage
20
40
60
80
age

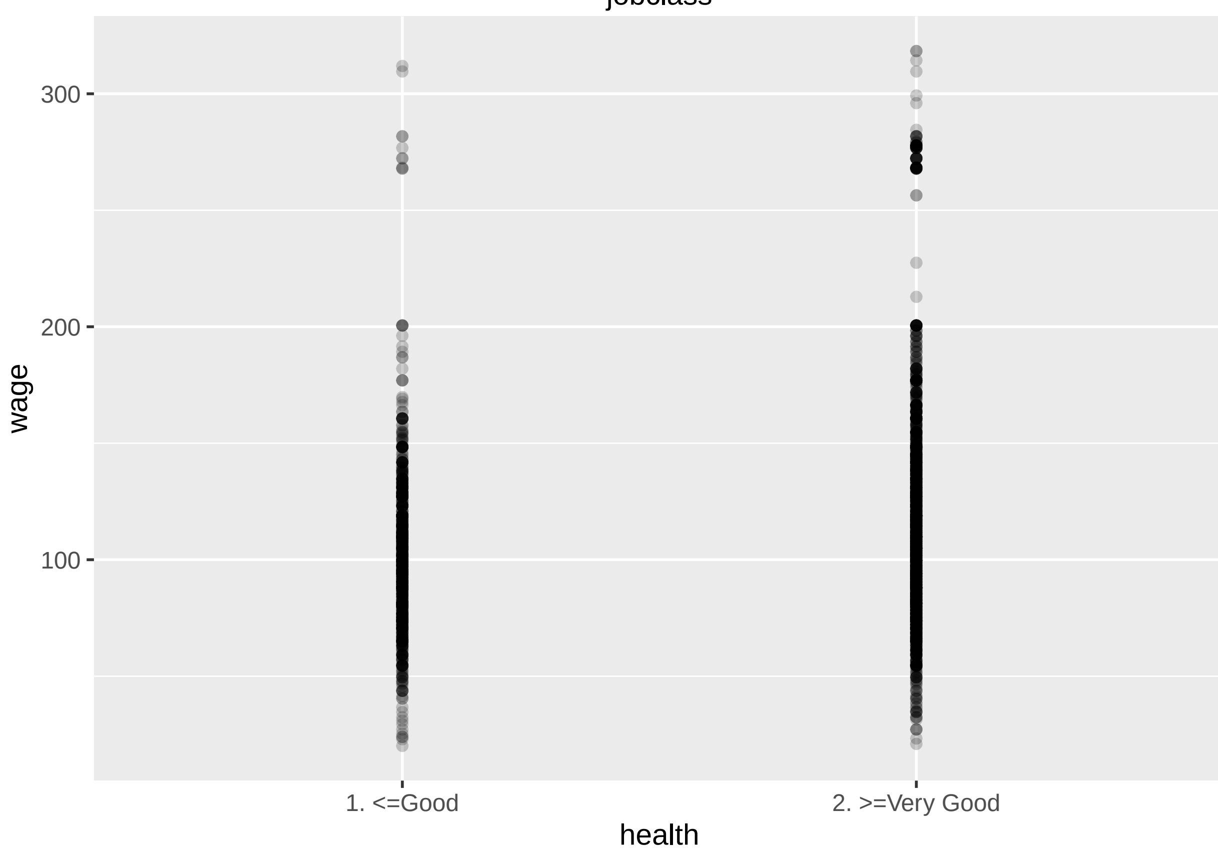

wage
300
200
100
1. Never Married
2. Married
3. Widowed
4. Divorced
5. Separated
maritl

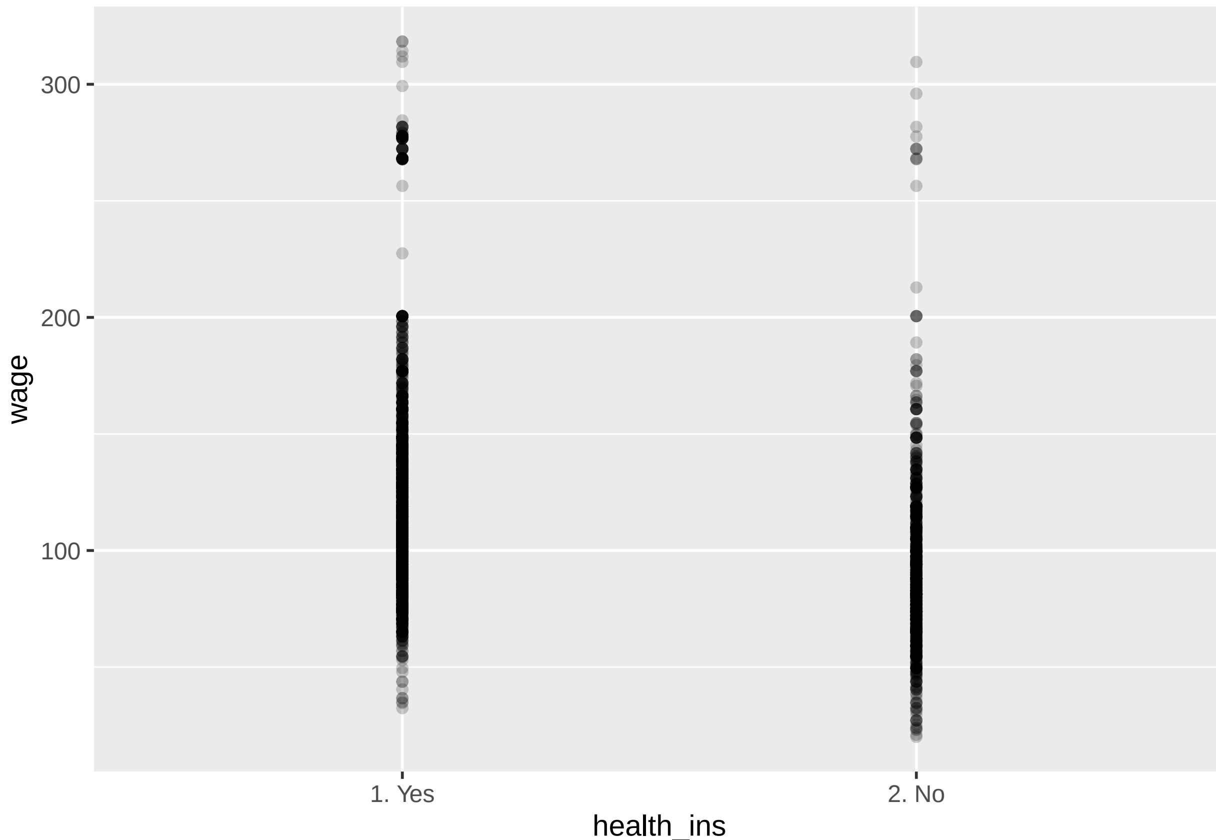

wage
300
200
100
1. White
2. Black
3. Asian
4. Other
race

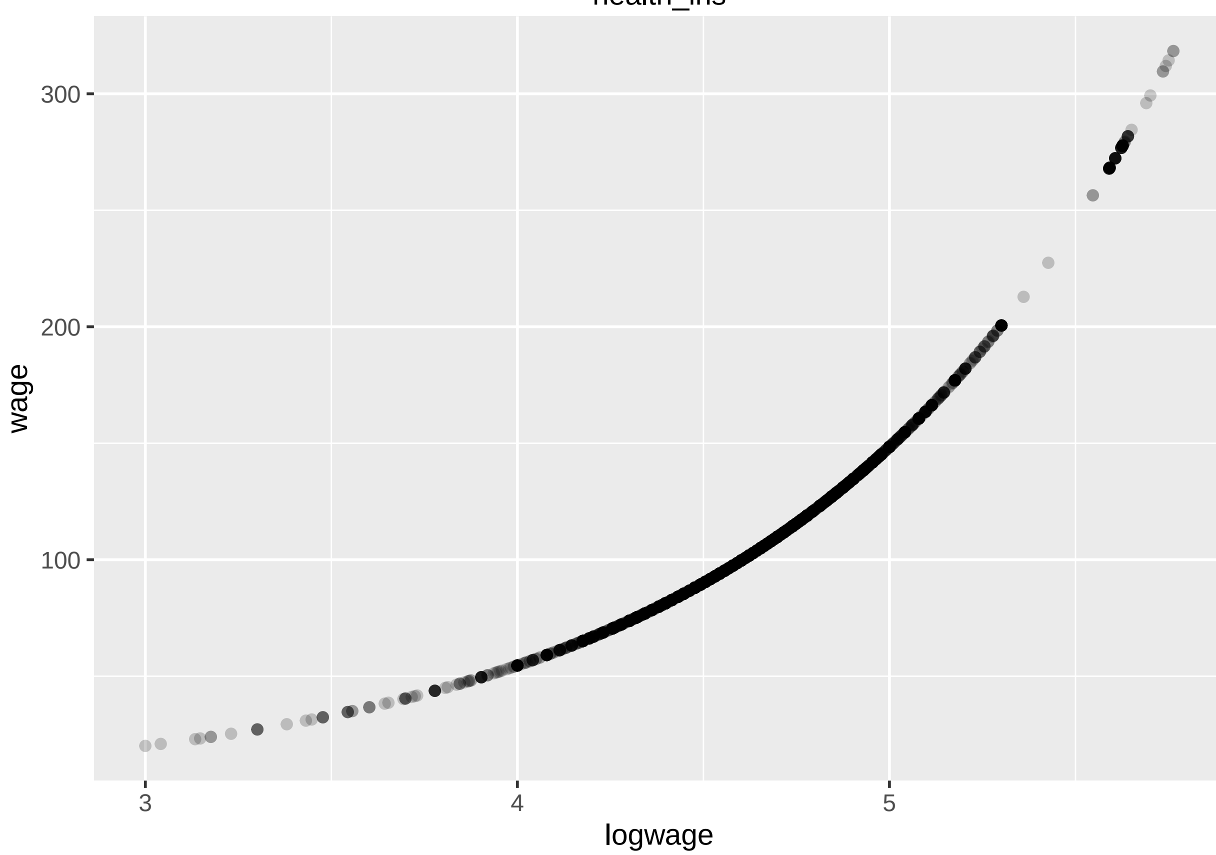

300
200
100
wage
1. < HS Grad
2. HS Grad
3. Some College
4. College Grad
5. Advanced Degree
education


300
200
100
wage
2. Middle Atlantic
region


wage
300
200
100
1. Industrial
2. Information
jobclass


wage
300
200
100
1. <=Good
2. >=Very Good
health


300
200
100
wage
1. Yes
2. No
health_ins


300
200
100
wage
3
4
5
logwage

```r
# Recipe
lm_pl_recipe <-
  recipe(formula = wage ~ age, data = Wage_train) %>%
  step_poly(age, degree = 2, options = list(raw = TRUE)) %>%
  step_normalize(all_predictors())

# Specification
lm_pl_spec <-
  linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

# Workflow
lm_pl_workflow <- workflow() %>%
  add_recipe(lm_pl_recipe) %>%
  add_model(lm_pl_spec)

# Finalize model
lm_pl_fit <- fit(lm_pl_workflow, data = Wage_train)

# Check RMSE
augment(lm_pl_fit, new_data = Wage_test) %>%
  rmse(truth = wage, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        39.8
```

```r
tidy(lm_pl_fit)
```

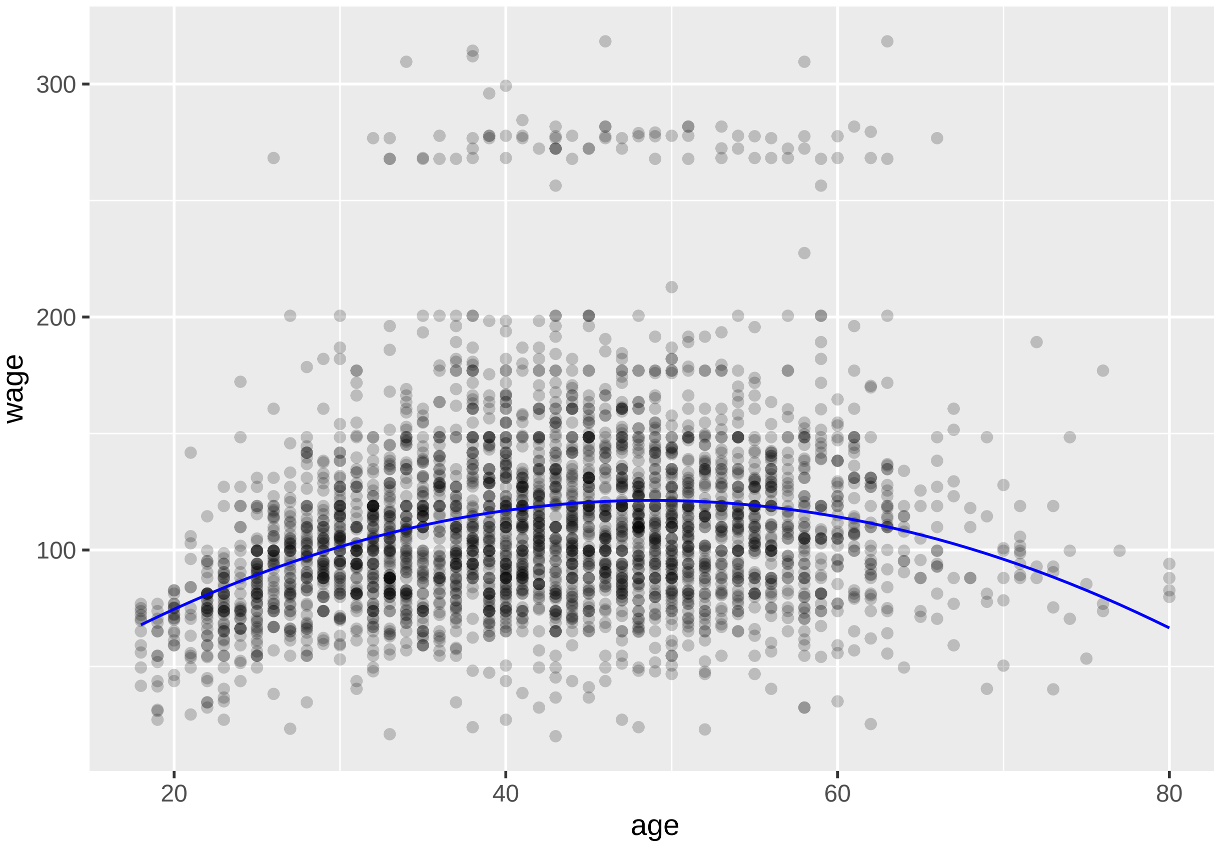
```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    112.       1.04    108.   0
## 2 age_poly_1      62.9      6.67      9.43 1.52e-20
## 3 age_poly_2     -56.1      6.67     -8.40 1.01e-16
```

```r
age_min <- min(Wage$age)
age_max <- max(Wage$age)
age_range <- tibble(age = seq(age_min, age_max))
age_range
```

```
## # A tibble: 63 x 1
##      age
##    <int>
##  1    18
##  2    19
##  3    20
##  4    21
##  5    22
##  6    23
##  7    24
##  8    25
##  9    26
```

```
## 10    27
## # ... with 53 more rows
```

```
regression_lines <- bind_cols(
  predict(lm_pl_fit, new_data = age_range),
  age_range
)

Wage %>%
  ggplot(aes(age, wage)) +
  geom_point(alpha = 0.2) +
  geom_line(aes(y = .pred), data = regression_lines, color = "blue")
```



All the other variables are categorical. Hot encoding these variables basically results in a "piecewise non-linear" function. However, exploring this most likely is not very interesting. Therefor we will now test splines also on the relationships of age and wage.

```
min(Wage$age)
```

```
## [1] 18
```

```
spline_recipe <- recipe(formula = wage ~ age, data = Wage_train) %>%
  step_bs(age, options = list(knots = 20, 30, 40, 50, 60, 70))

# Specification
spline_spec <-
  linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")
```

```r
# Workflow
spline_workflow <- workflow() %>%
  add_recipe(spline_recipe) %>%
  add_model(spline_spec)

# Finalize model
spline_fit <- fit(spline_workflow, data = Wage_train)

# Check RMSE
augment(spline_fit, new_data = Wage_test) %>%
  rmse(truth = wage, estimate = .pred)
```

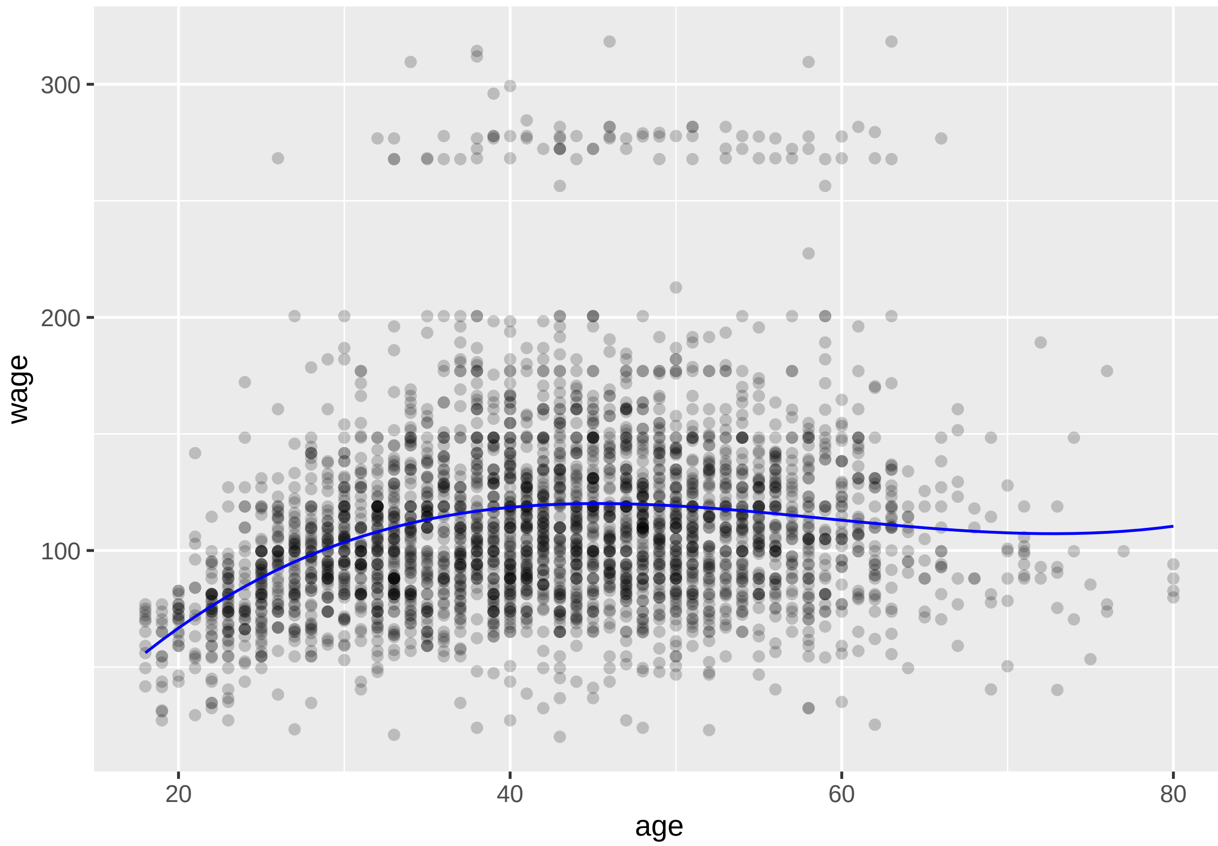
```
## Warning in bs(x = c(18L, 24L, 43L, 30L, 45L, 34L, 51L, 50L, 56L, 40L, 49L, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        39.7
```

```r
regression_lines <- bind_cols(
  predict(spline_fit, new_data = age_range),
  age_range
)
```

```
## Warning in bs(x = 18:80, degree = 3L, knots = numeric(0), Boundary.knots =
## c(18L, : some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```r
Wage %>%
  ggplot(aes(age, wage)) +
  geom_point(alpha = 0.2) +
  geom_line(aes(y = .pred), data = regression_lines, color = "blue")
```



Both the polynomial regression as well as splines capture the relationship between age and wage better than a simple linear regression. We can see that the splines fit the data even better than the polynomial regression, especially at the border regions.