Mission Hospital Case: Workbook

Robert MD Bui

```
setwd("G:/My Drive/academics/2122_uic_ms/2022_spring/ids506_health/mission")

library(tidyverse) |> suppressPackageStartupMessages()
library(tidymodels) %>% suppressPackageStartupMessages()
library(patchwork) %>% suppressPackageStartupMessages()
library(kableExtra) %>% suppressPackageStartupMessages()
library(Cairo) %>% suppressPackageStartupMessages()
```

Cleaning

```
# Loading custom scripts
source("scripts/clean_data.R")
source("scripts/na_table.R")

df <- clean_data(read_csv("data/sheet4.csv",show_col_types = FALSE))
na_table(df)
```

```
##                       prop_NA
## meas_CREATININE    0.133064516
## body_bphigh        0.092741935
## body_bplow         0.092741935
## meas_UREA          0.052419355
## meas_HB            0.008064516
## id                 0.000000000
## demo_age           0.000000000
## demo_female        0.000000000
## demo_unmarried     0.000000000
## code_ACHD          0.000000000
## code_CAD_DVD       0.000000000
## code_CAD_SVD       0.000000000
## code_CAD_TVD       0.000000000
## code_CAD_VSD       0.000000000
## code_OS_ASD        0.000000000
## code_PM_VSD        0.000000000
## code_RHD           0.000000000
## code_othr_heart    0.000000000
## code_othr_respi    0.000000000
## code_othr_genrl    0.000000000
## code_othr_nerve    0.000000000
## code_othr_terta    0.000000000
## body_wgt           0.000000000
```

```
## body_hgt           0.000000000
## body_hrpulse       0.000000000
## body_rr            0.000000000
## hist_diabetes1     0.000000000
## hist_diabetes2     0.000000000
## hist_hypertension1 0.000000000
## hist_hypertension2 0.000000000
## hist_hypertension3 0.000000000
## hist_other         0.000000000
## arr_walkin         0.000000000
## arr_ambulance      0.000000000
## arr_transfer       0.000000000
## is_alert           0.000000000
## is_emergency       0.000000000
## cost               0.000000000
## cost_ln            0.000000000
## stay_total         0.000000000
## stay_icu           0.000000000
## stay_ward          0.000000000
## implant            0.000000000
## implant_cost       0.000000000
```

```r
# write_csv(df,"data/data.csv")
# write_csv(df_naomit,"data/data_naomit.csv")
```

Exploratory Analysis

```r
######################################## DISCRETE PLOTTING FUNCTION ##############################
plot_discrete <- function(dframe,ycol = "y",col){
  pbase <- dframe %>%
    ggplot()+
    aes(x = !!sym(ycol),
        fill = !!sym(col))+
    scale_fill_viridis_d()+
    theme_minimal()

  (pbase +
      geom_histogram(bins = 10,position="fill") +
      theme(legend.position = "none")) | (pbase + geom_histogram(bins = 10))
}

######################################## CONTINUOUS PLOTTING FUNCTION ##############################
plot_continuous <- function(dframe,ycol = "y",col){
  pbase <- dframe %>%
    ggplot()+
    aes(y = !!sym(ycol),
        x = !!sym(col))+
    geom_point(color = "#440154")+
    geom_smooth(color = "#fde725", method = "lm")+
    scale_fill_viridis_d()+
    theme_minimal()
```

```
  return(pbase)
}
```

```
##################################### DRAW ALL PLOTS ########################################
index <- df %>%
  select(-c(id,cost,cost_ln)) %>%
  sapply(typeof) %>%
  as.data.frame() %>%
  rownames_to_column("colname")

colnames(index) <- c("colname","coltype")

for(i in (1:nrow(index))){
  name = index[i,1]
  type = index[i,2]

  if(type %in% c("double","numeric","integer")){
    plt.title <- paste("Figure ",i,": Plotting ",name, " against Target Variable",sep = "")
    print(plot_continuous(df,"cost",name)+ggtitle(plt.title))
  }else if(type %in% c("factor","logical")){
    plt.title <- paste("Figure ",i,": Plotting ",name, " against Target Variable",sep = "")
    print(plot_discrete(df,"cost",name)+plot_annotation(title = plt.title))
  }
}
```

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

Figure 1: Plotting demo_age against Target Variable

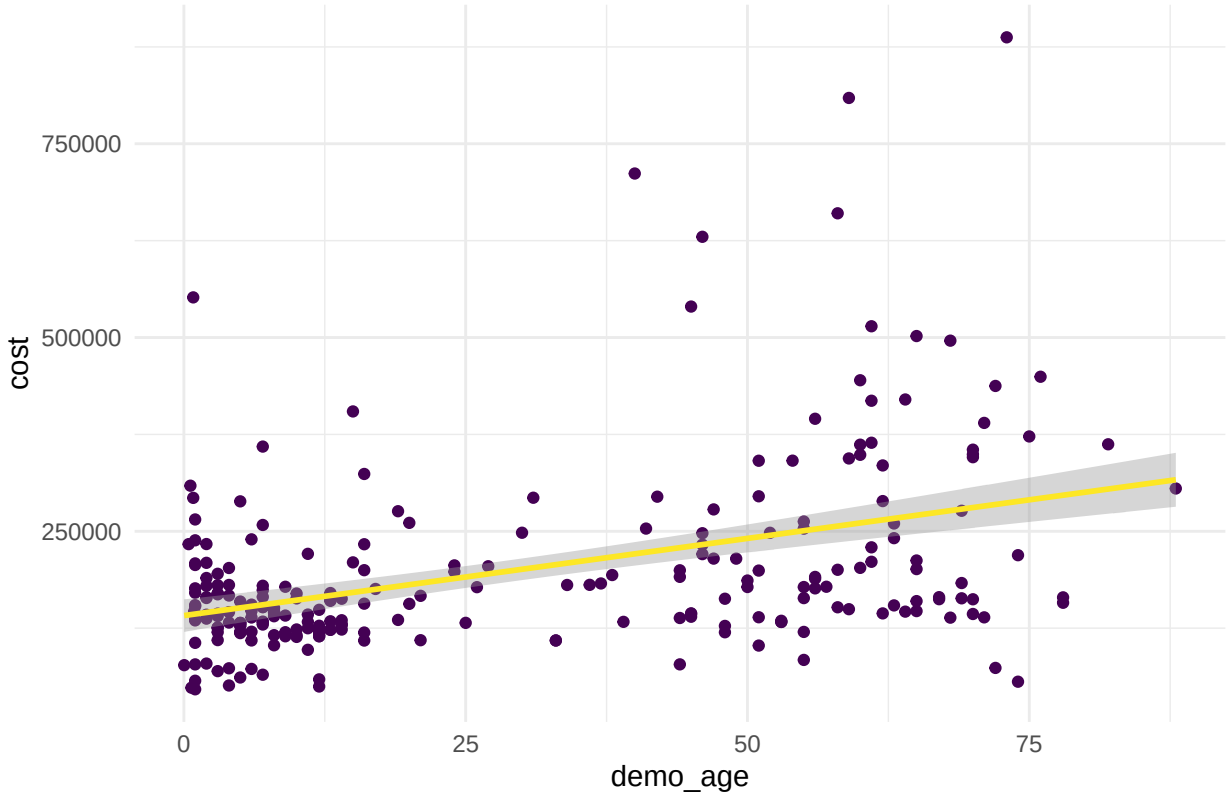


Figure 2: Plotting demo_female against Target Variable

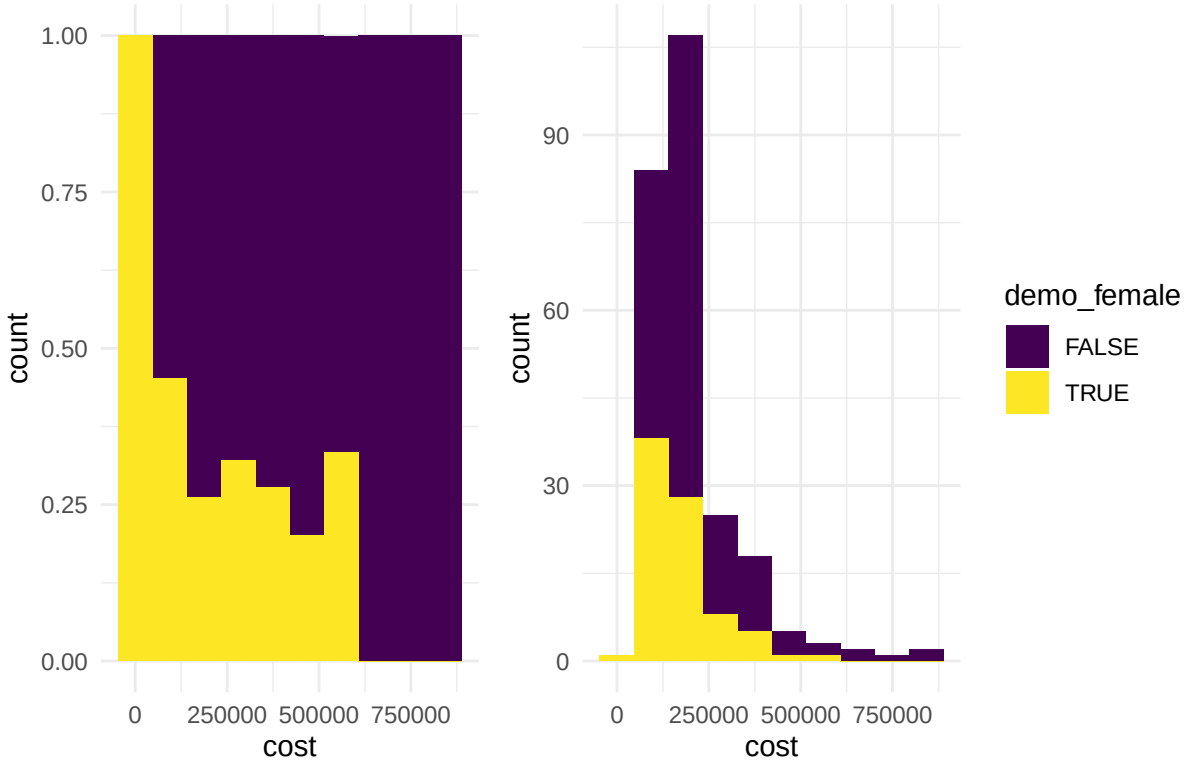


Figure 3: Plotting demo_unmarried against Target Variable

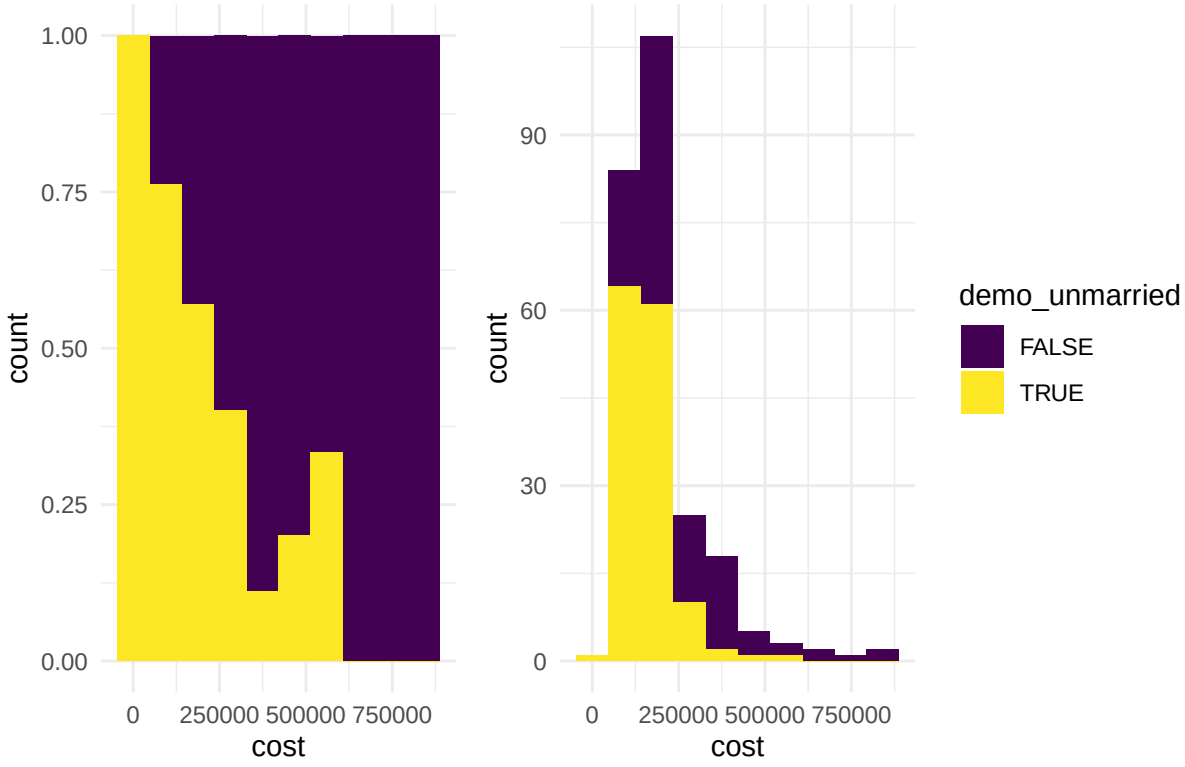


Figure 4: Plotting code_ACHD against Target Variable

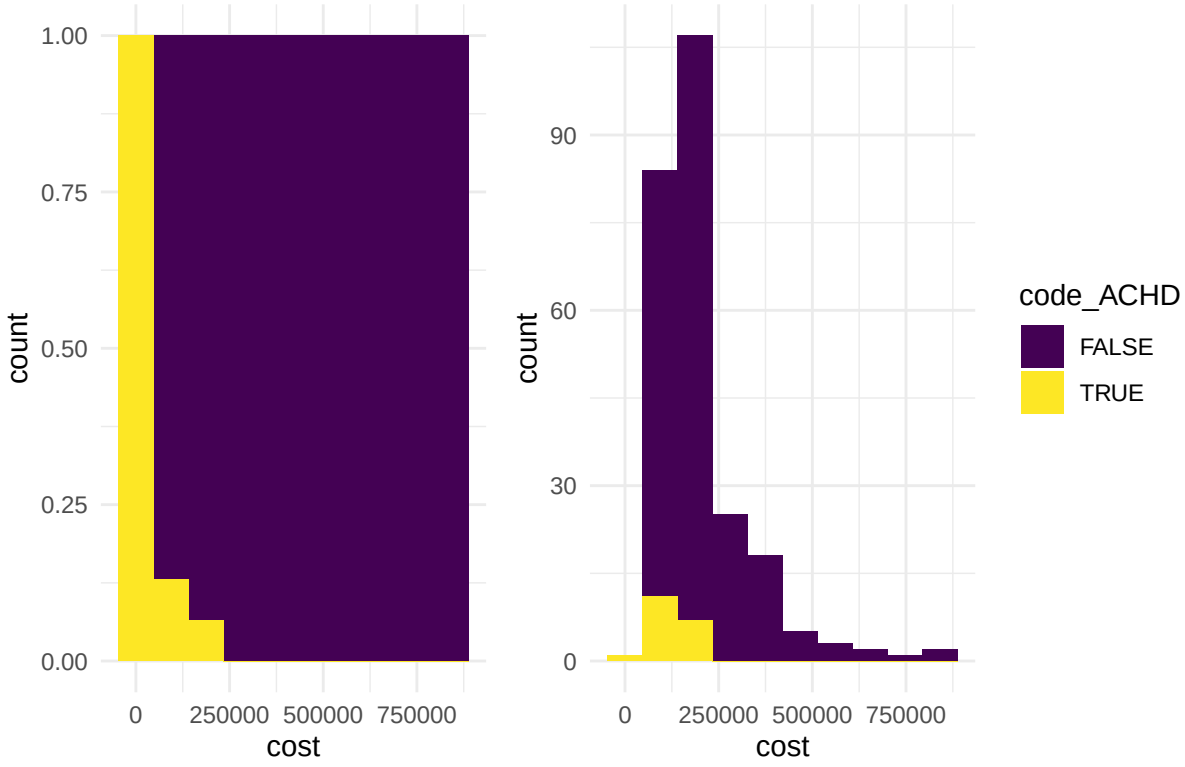


Figure 5: Plotting code_CAD_DVD against Target Variable

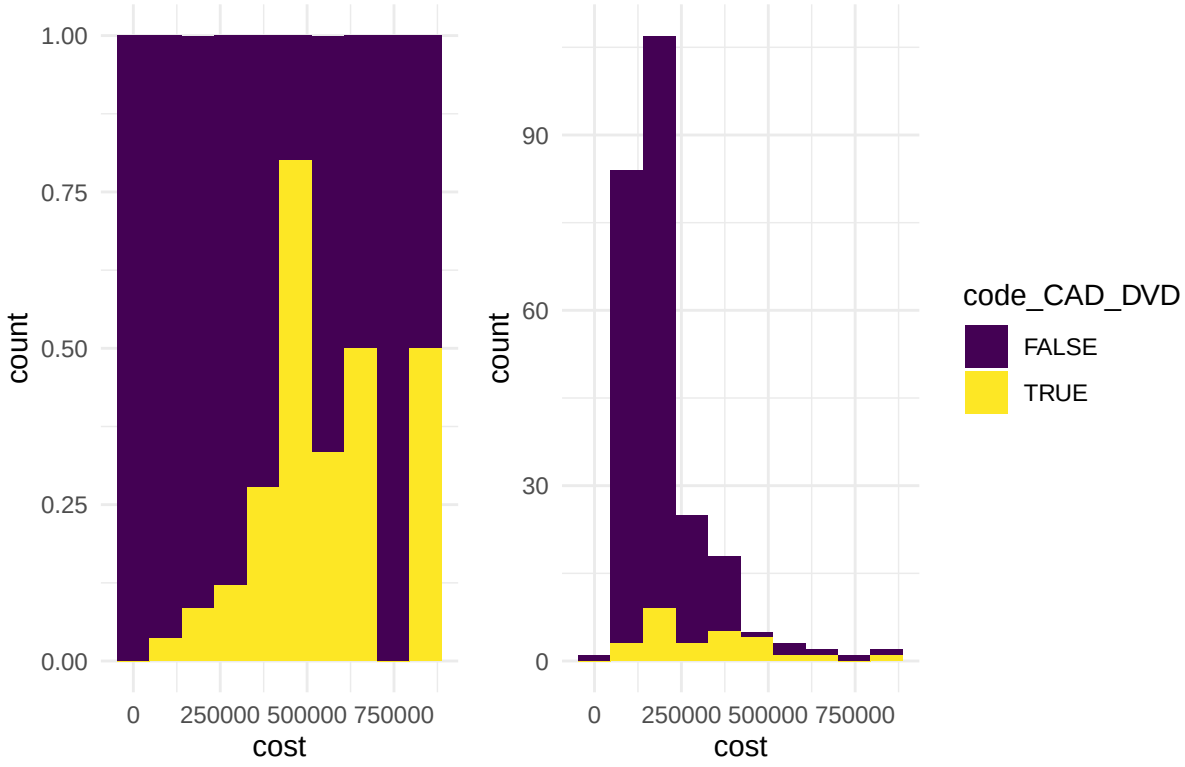


Figure 6: Plotting code_CAD_SVD against Target Variable

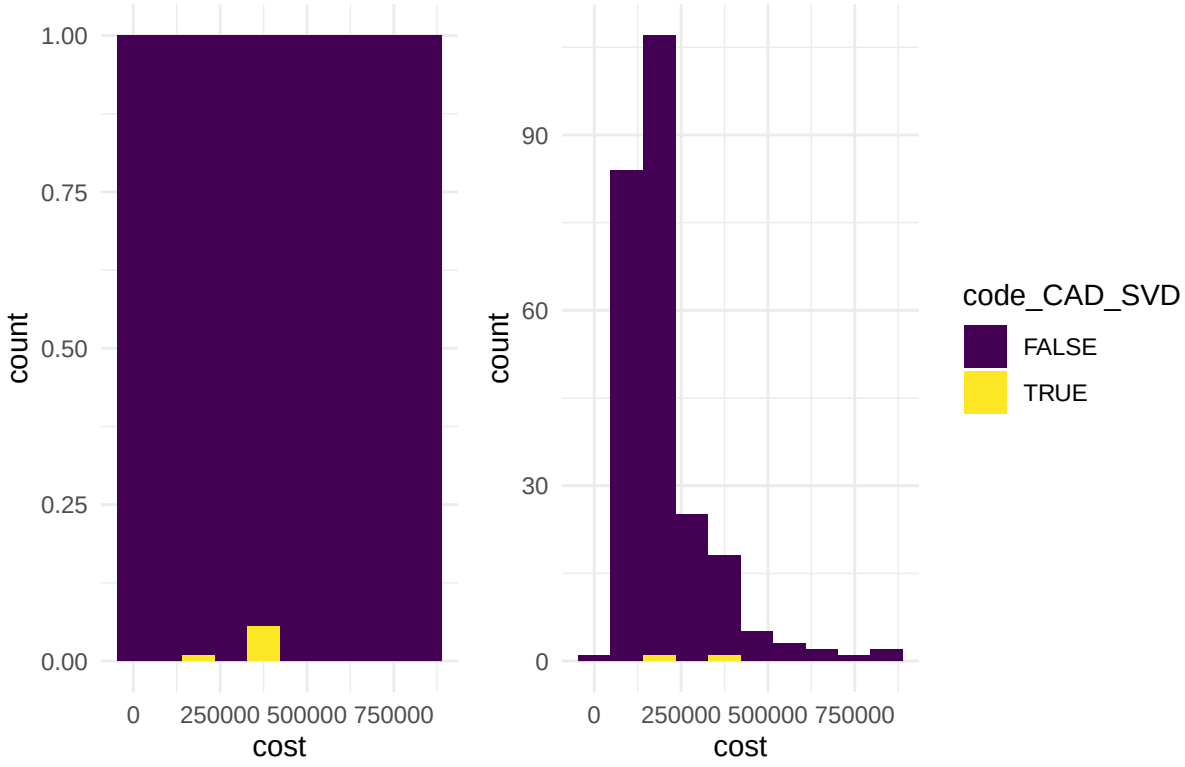


Figure 7: Plotting code_CAD_TVD against Target Variable

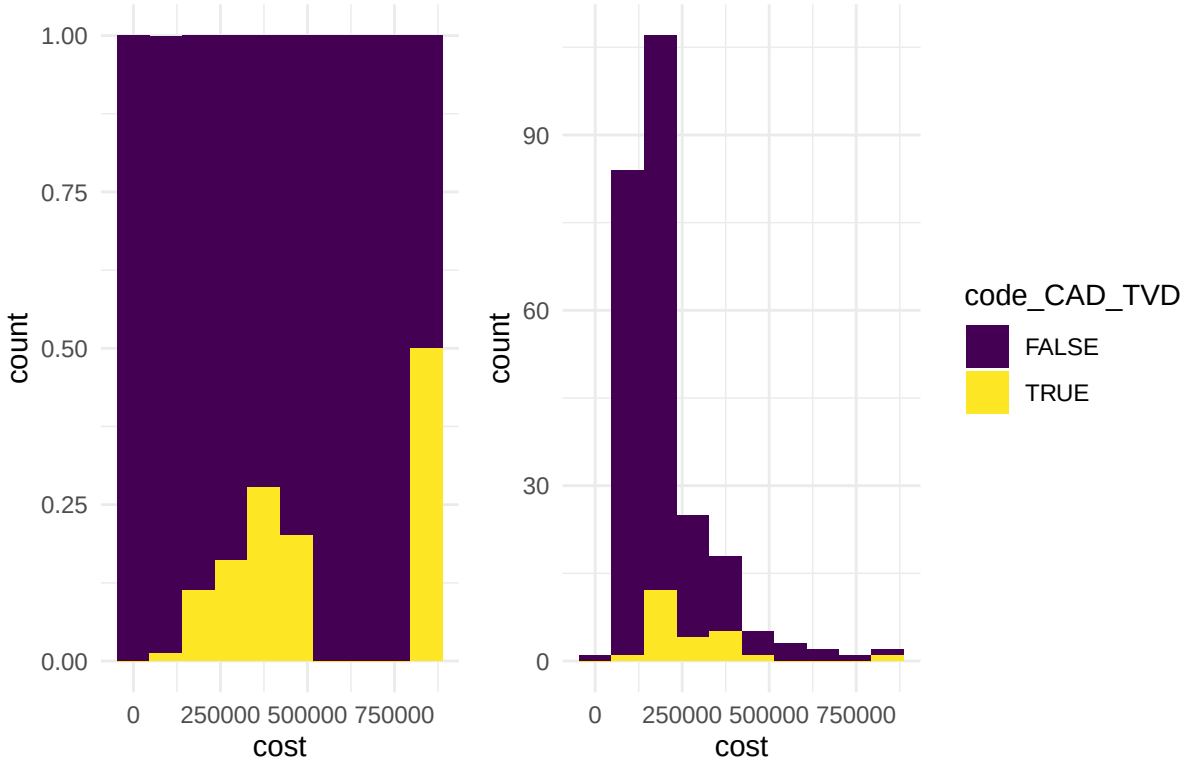


Figure 8: Plotting code_CAD_VSD against Target Variable

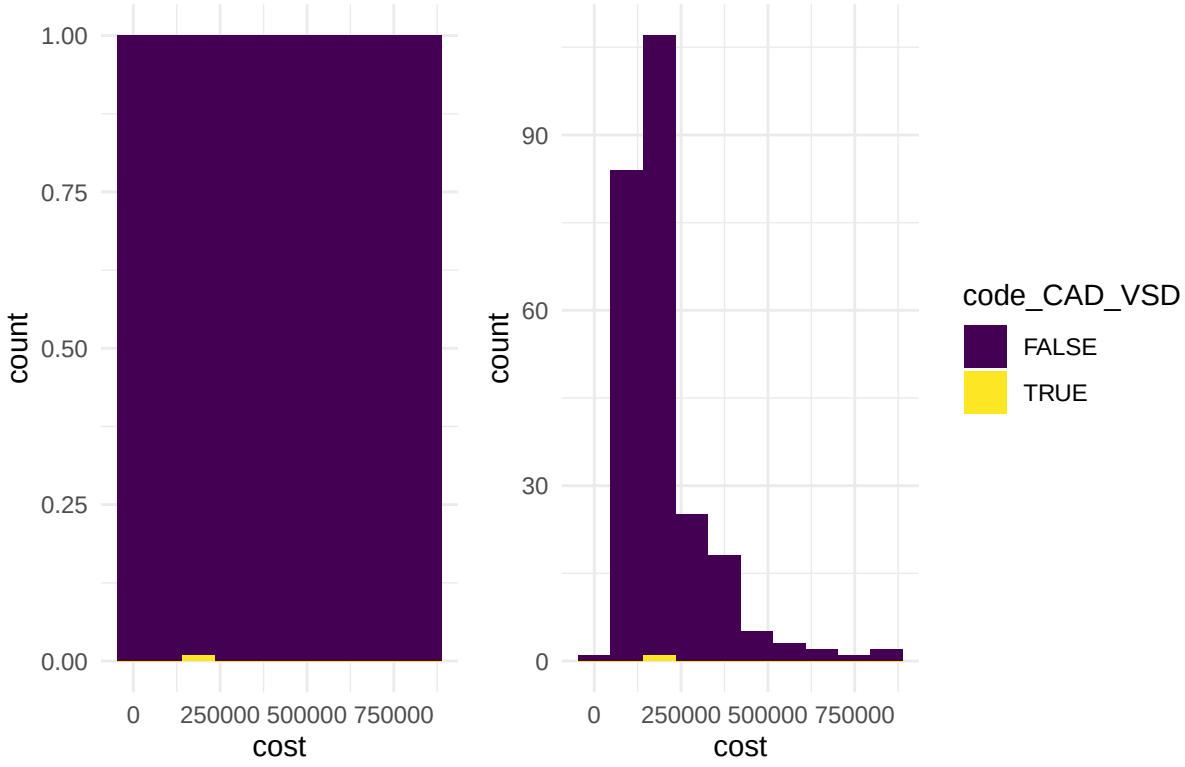


Figure 9: Plotting code_OS_ASD against Target Variable

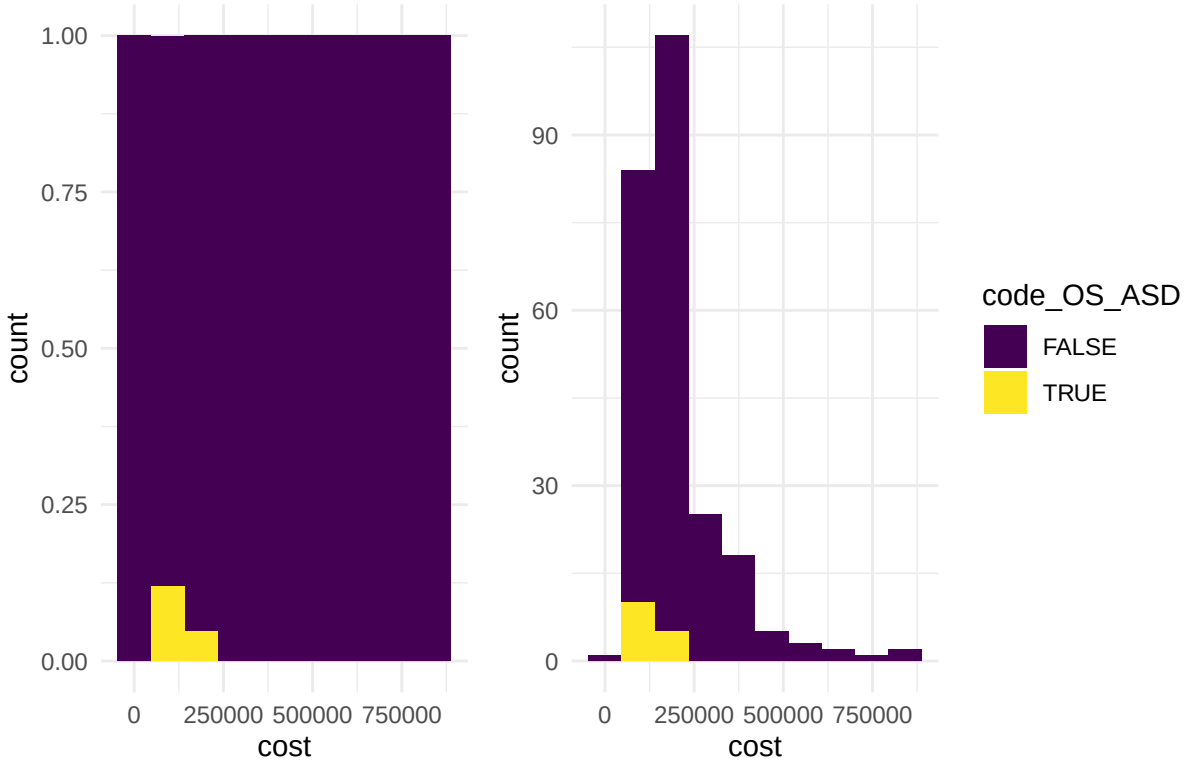


Figure 10: Plotting code_PM_VSD against Target Variable

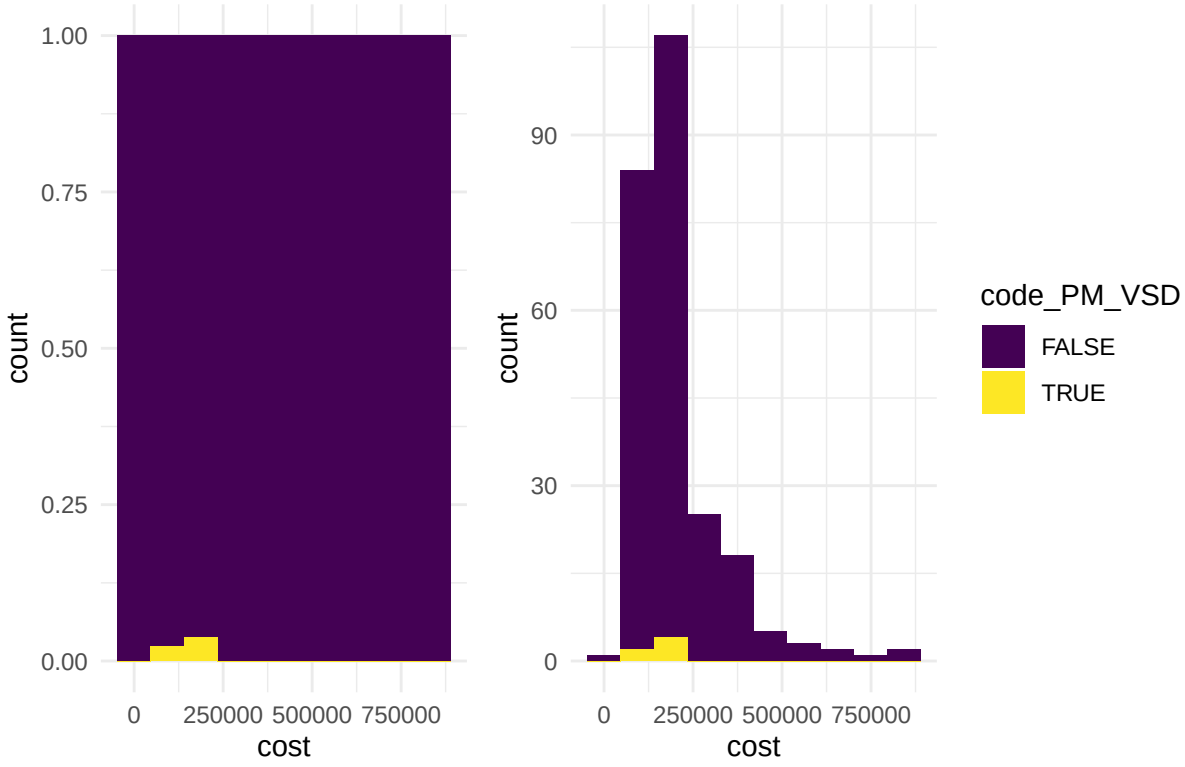


Figure 11: Plotting code_RHD against Target Variable

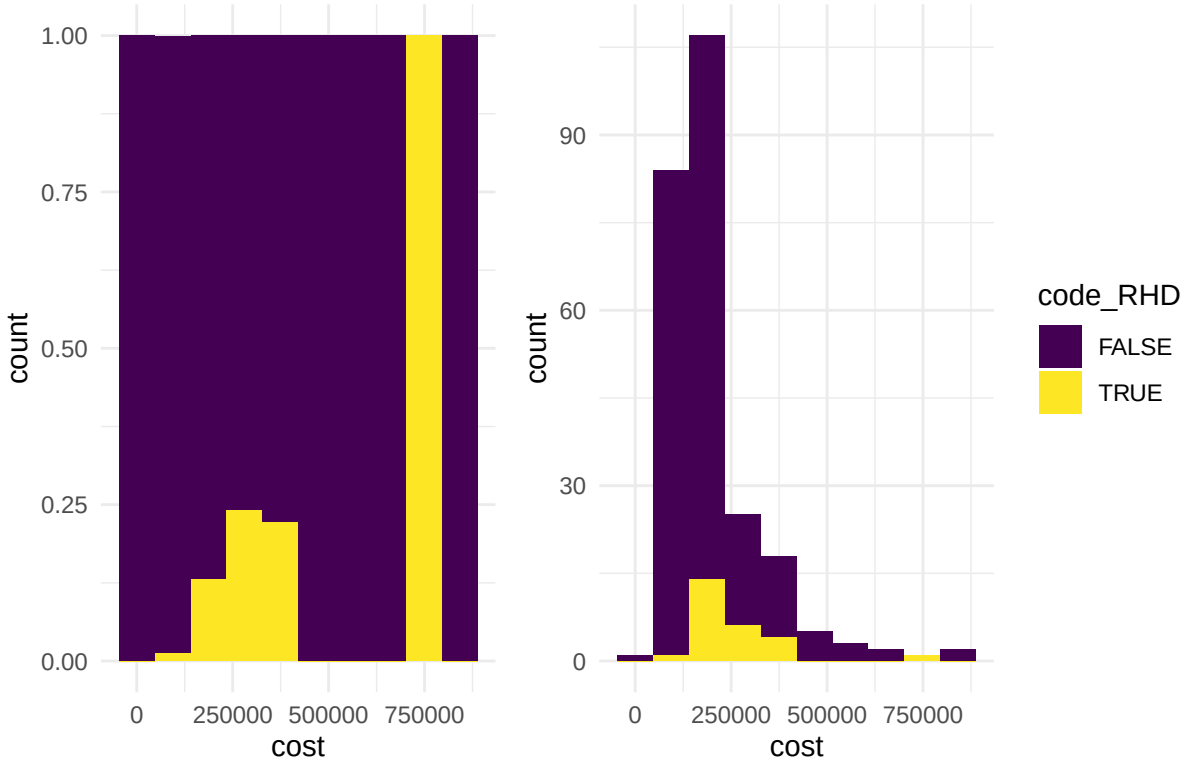


Figure 12: Plotting code_othr_heart against Target Variable

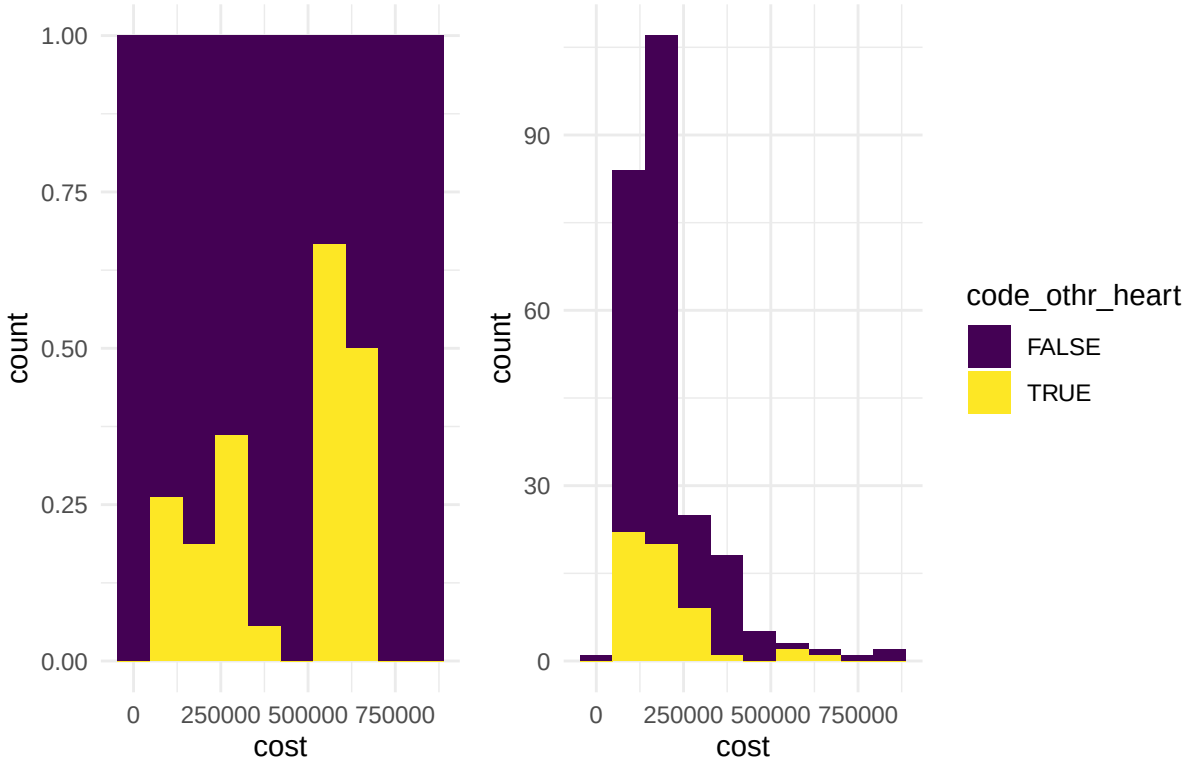


Figure 13: Plotting code_othr_respi against Target Variable

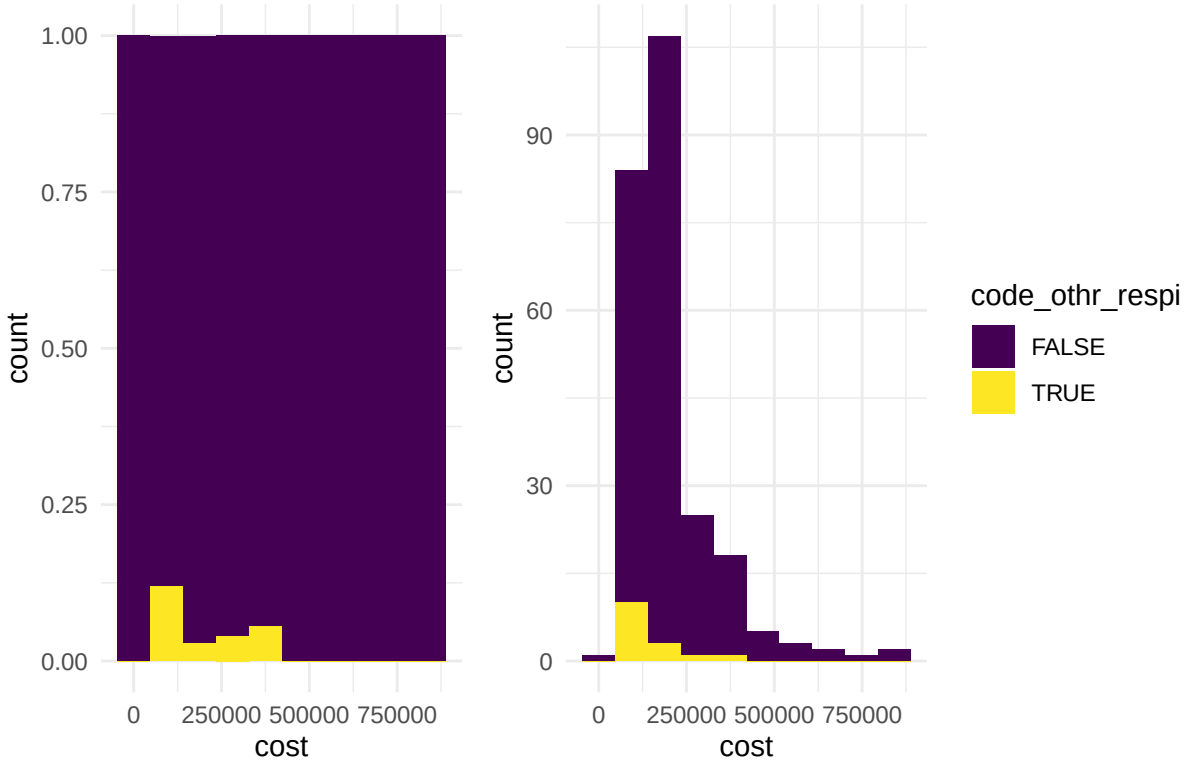


Figure 14: Plotting code_othr_genrl against Target Variable

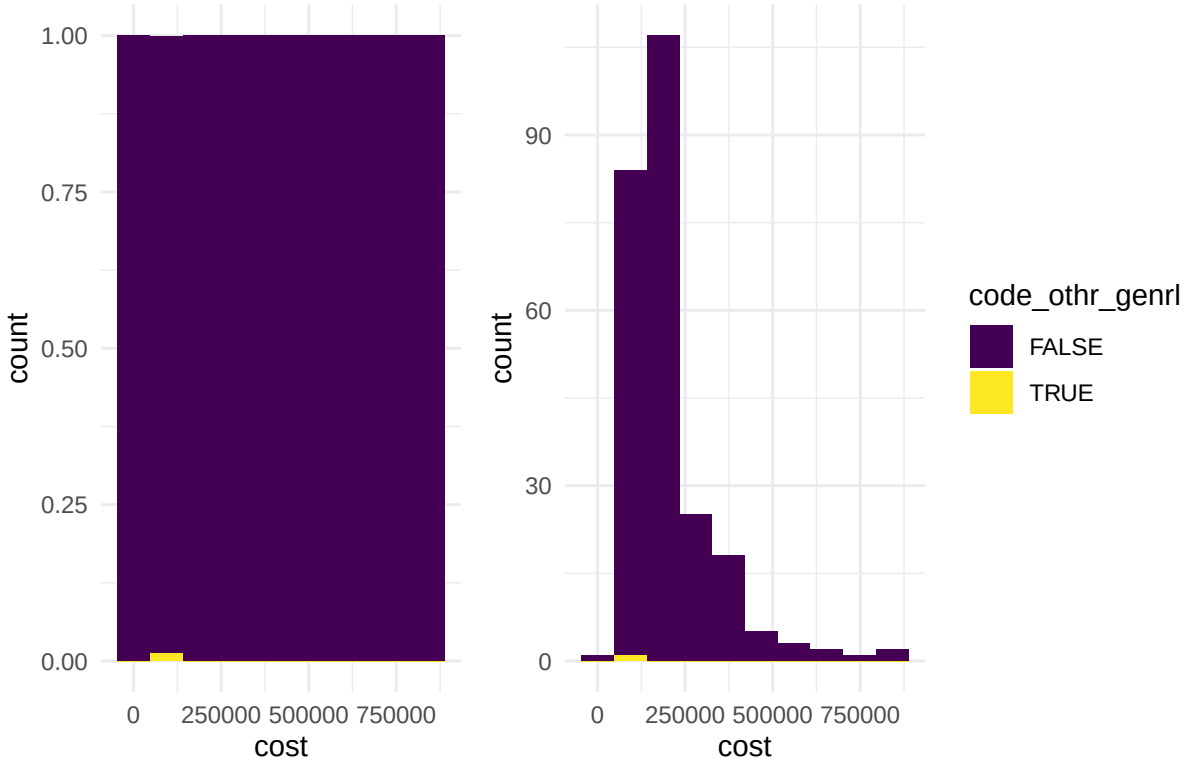


Figure 15: Plotting code_othr_nerve against Target Variable

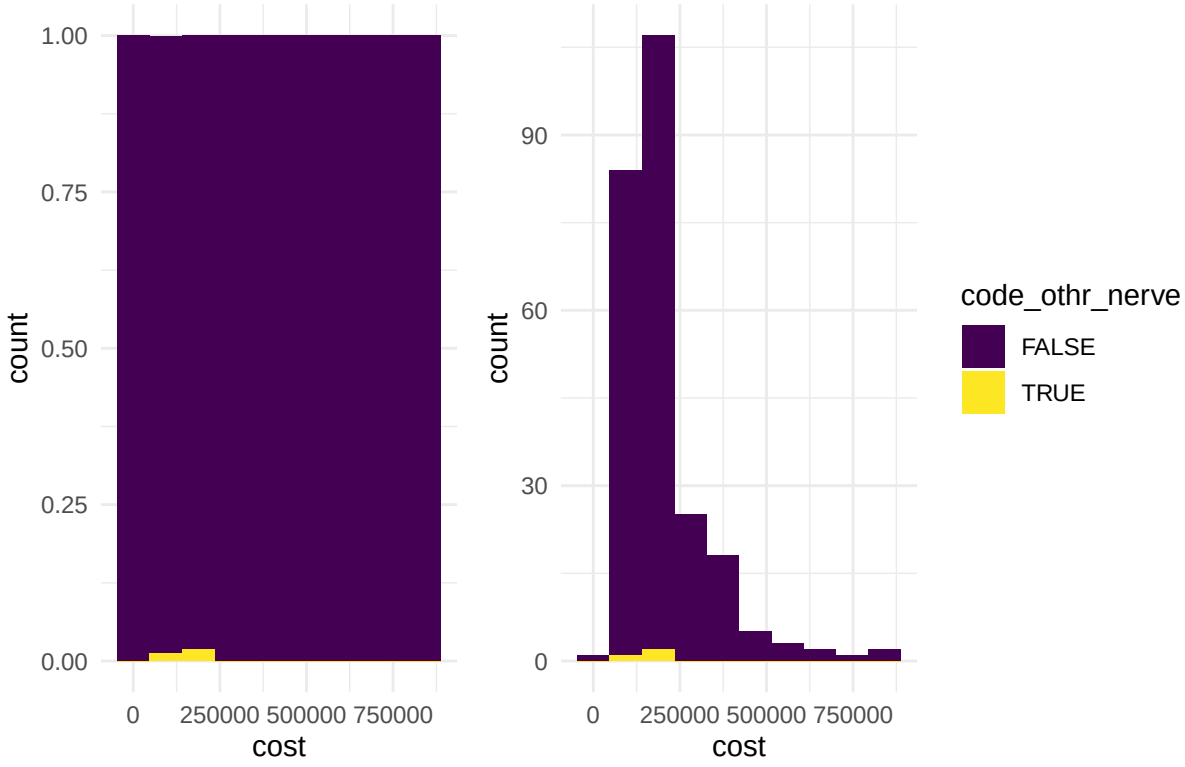
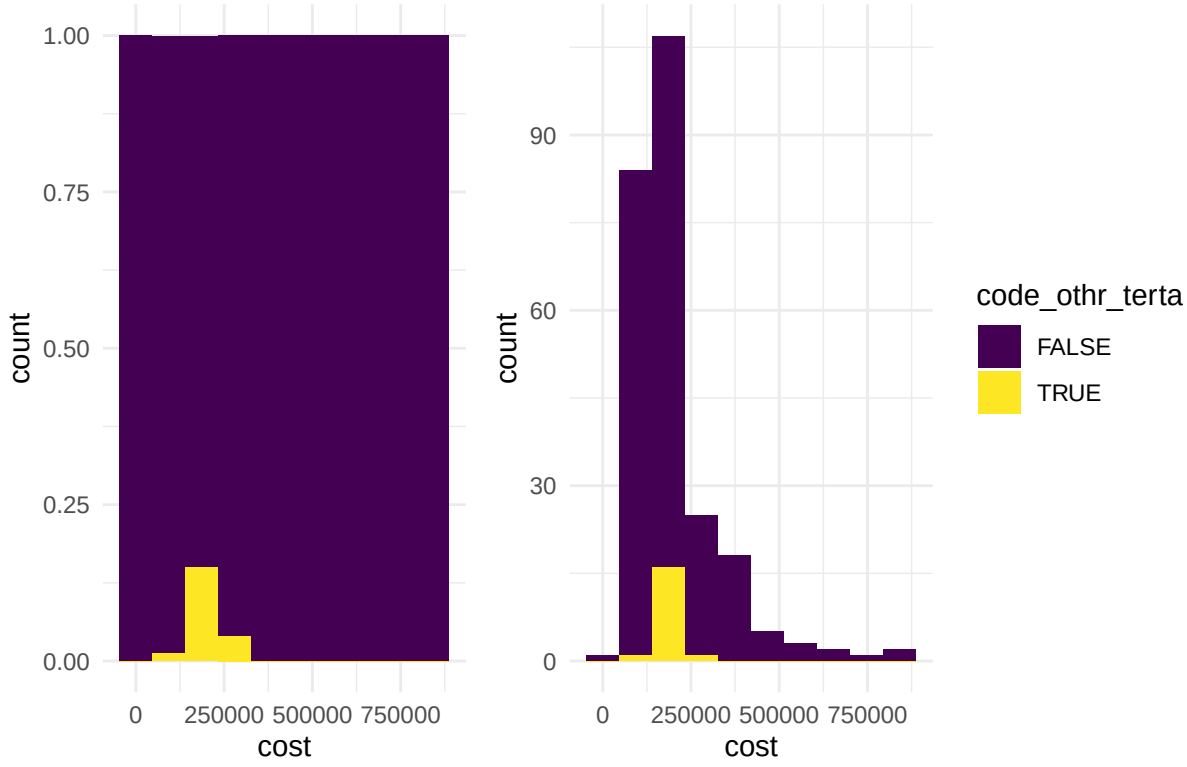
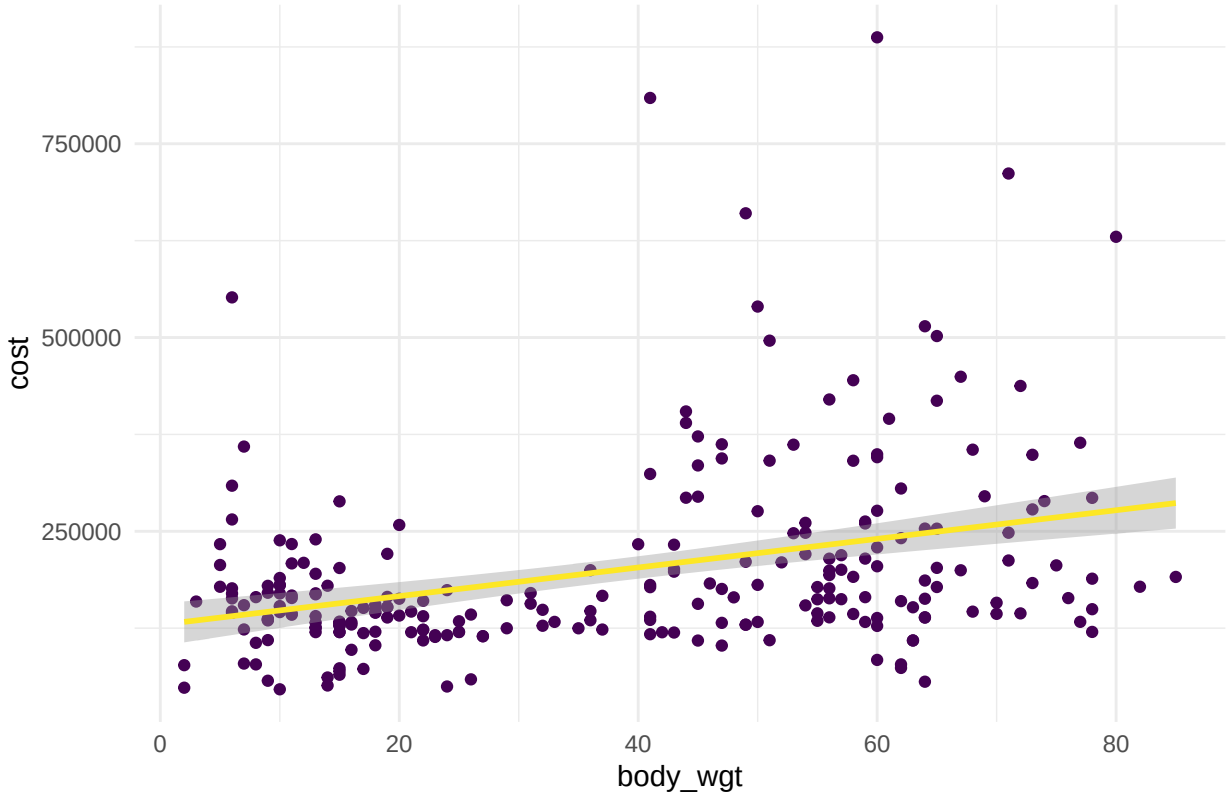


Figure 16: Plotting code_othr_terta against Target Variable



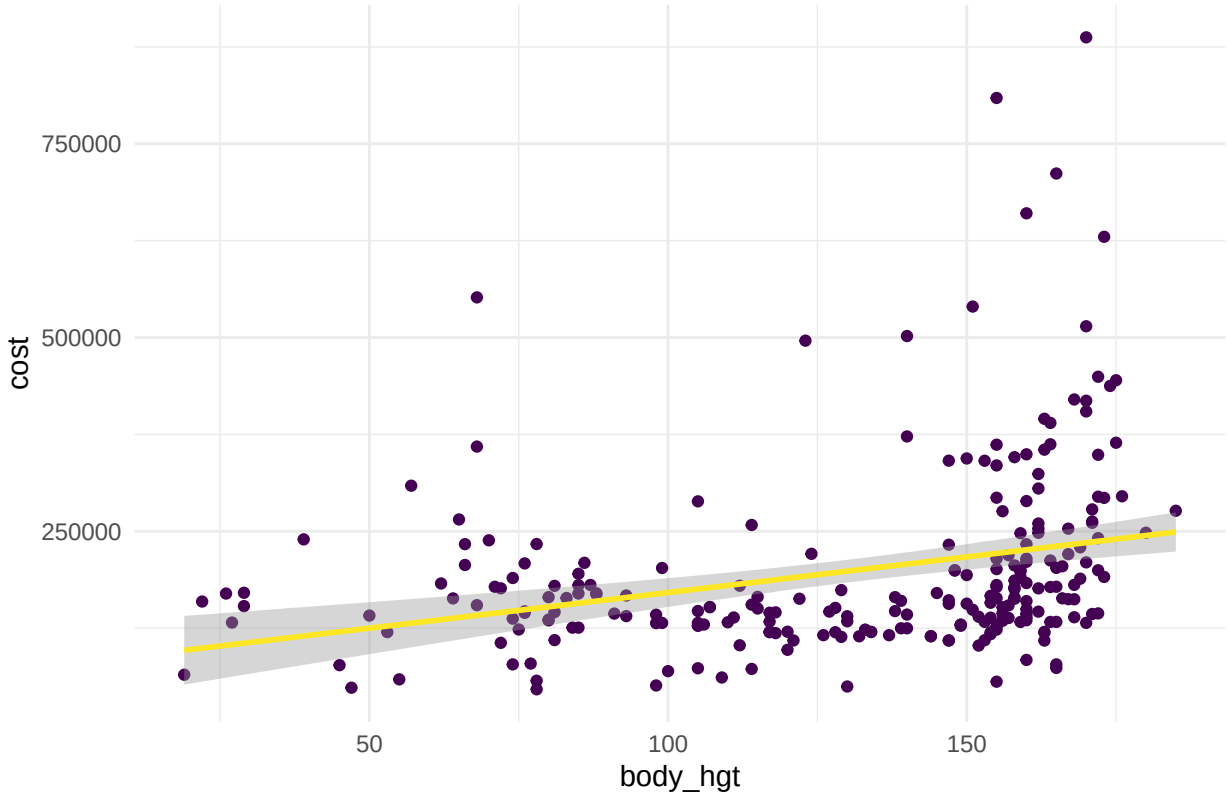

```
## `geom_smooth()` using formula 'y ~ x'
```

Figure 17: Plotting body_wgt against Target Variable



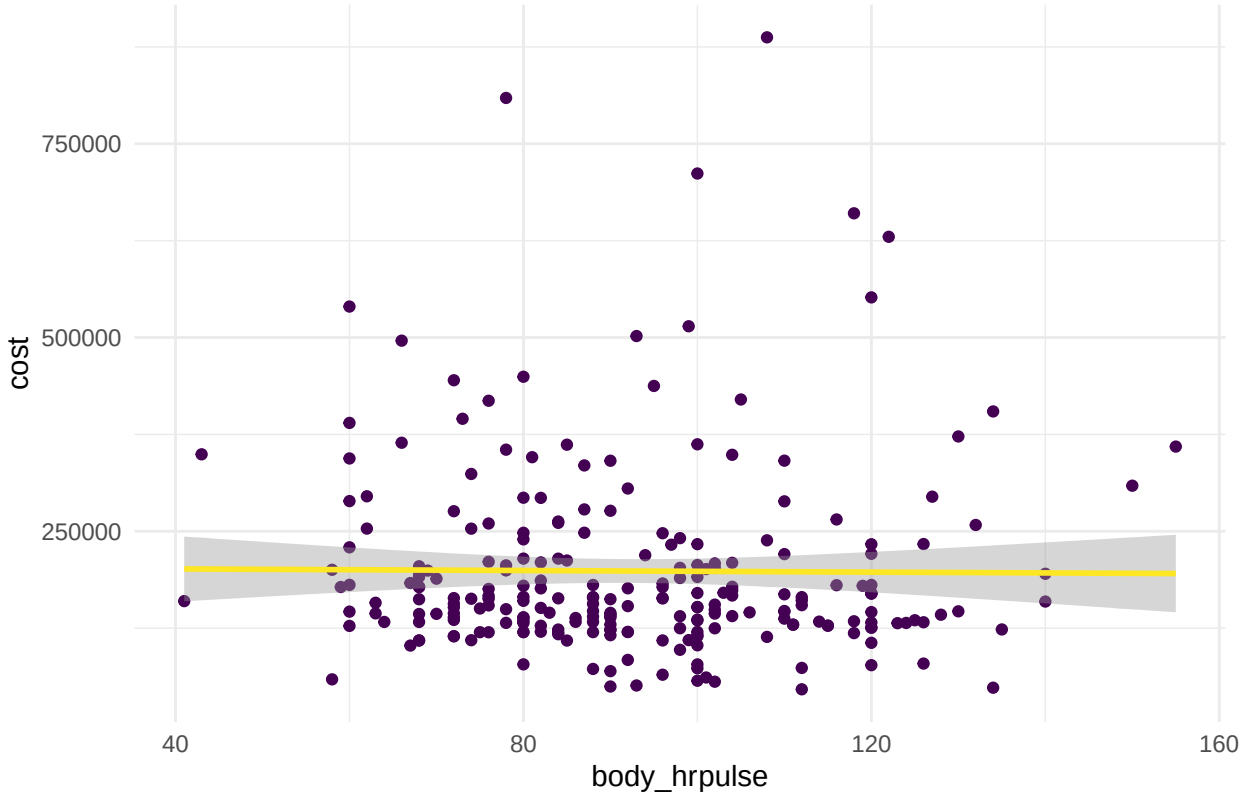

```
## `geom_smooth()` using formula 'y ~ x'
```

Figure 18: Plotting body_hgt against Target Variable




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

Figure 19: Plotting body_hrpulse against Target Variable

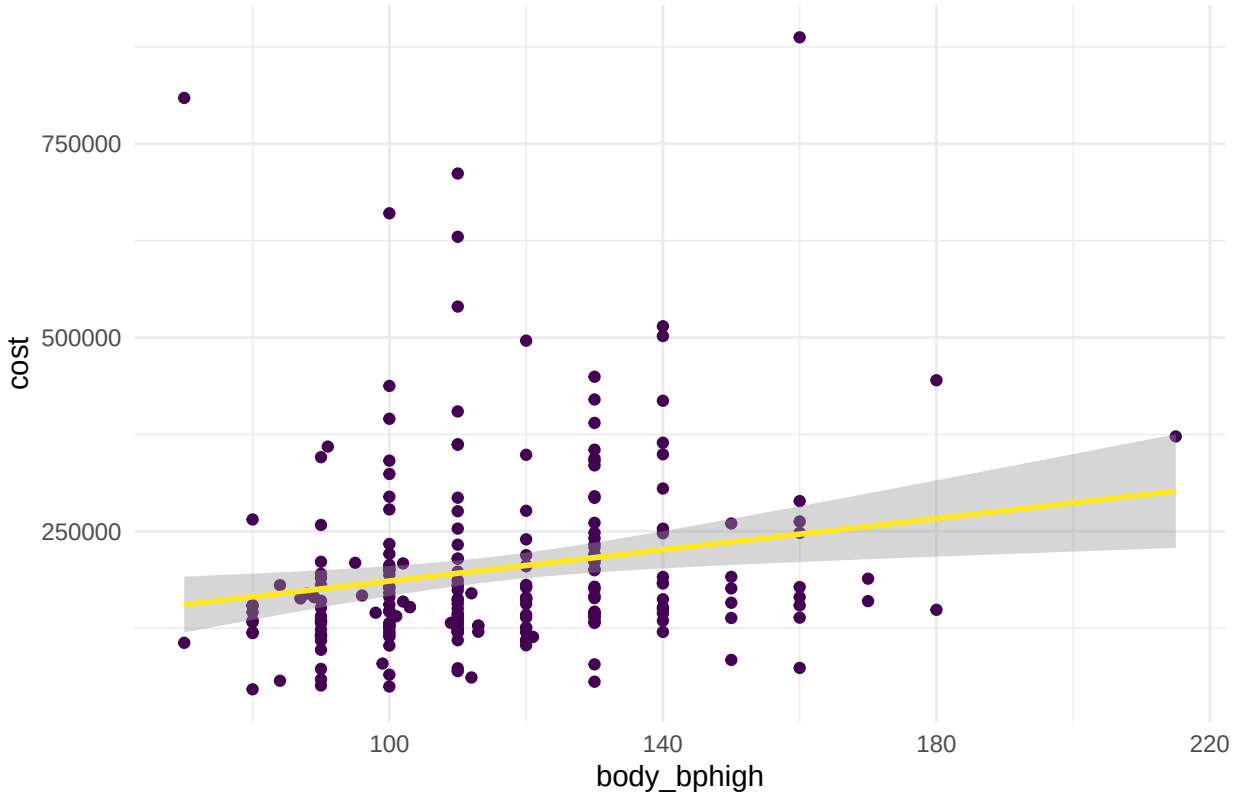



```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 23 rows containing non-finite values (stat_smooth).

## Warning: Removed 23 rows containing missing values (geom_point).
```

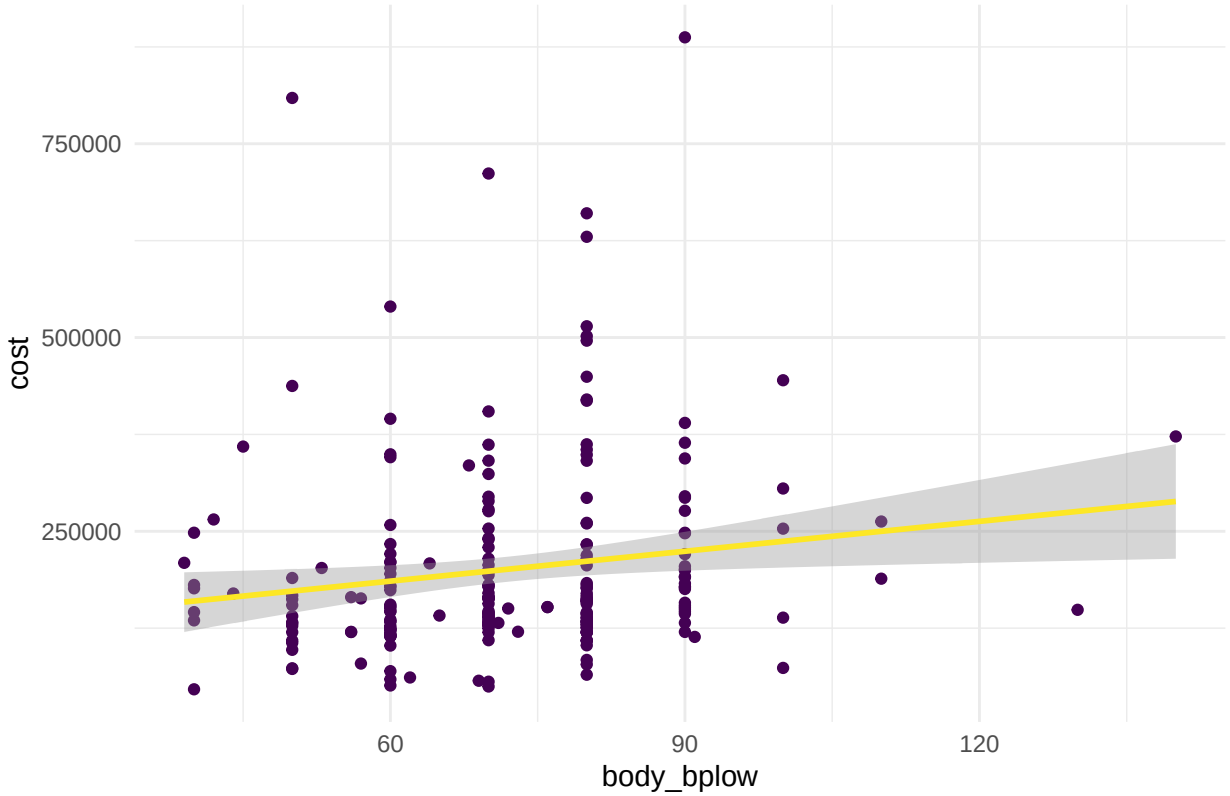
Figure 20: Plotting body_bphigh against Target Variable




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

## Warning: Removed 23 rows containing non-finite values (stat_smooth).
## Removed 23 rows containing missing values (geom_point).
```

Figure 21: Plotting body_bplow against Target Variable




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

Figure 22: Plotting body_rr against Target Variable

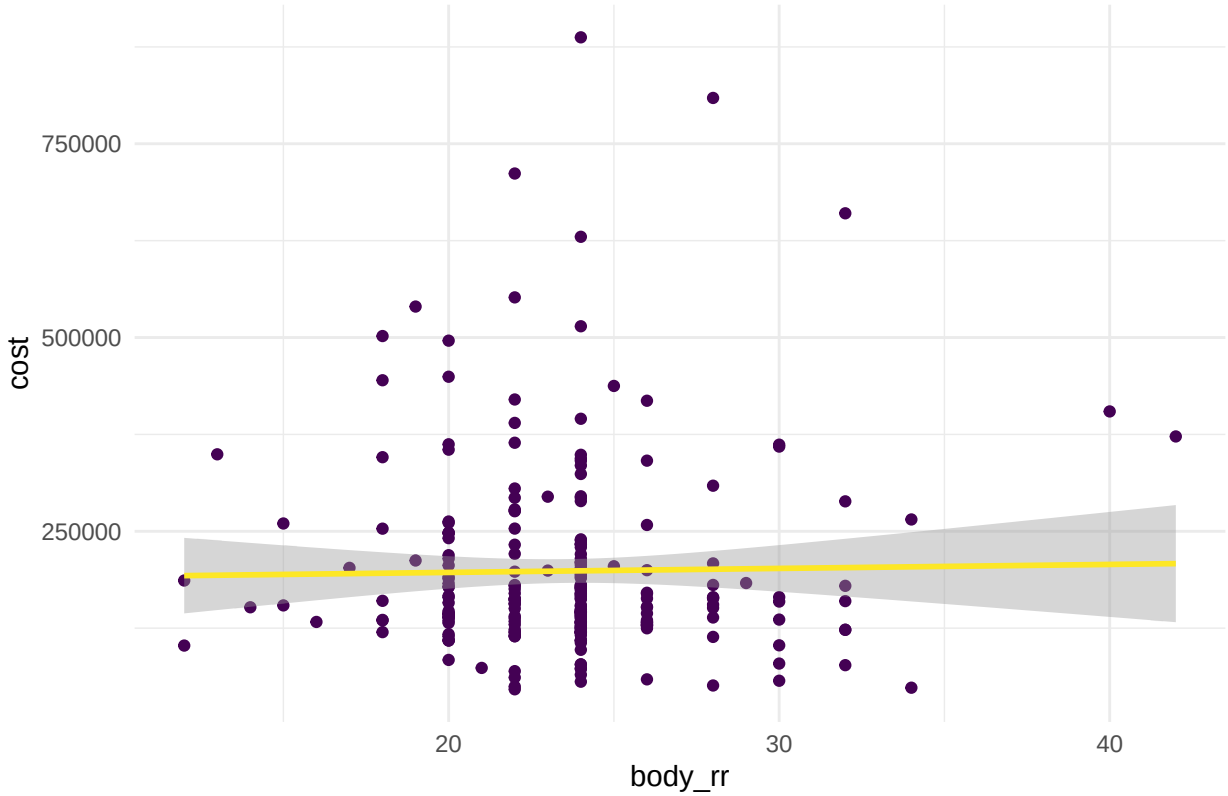


Figure 23: Plotting hist_diabetes1 against Target Variable

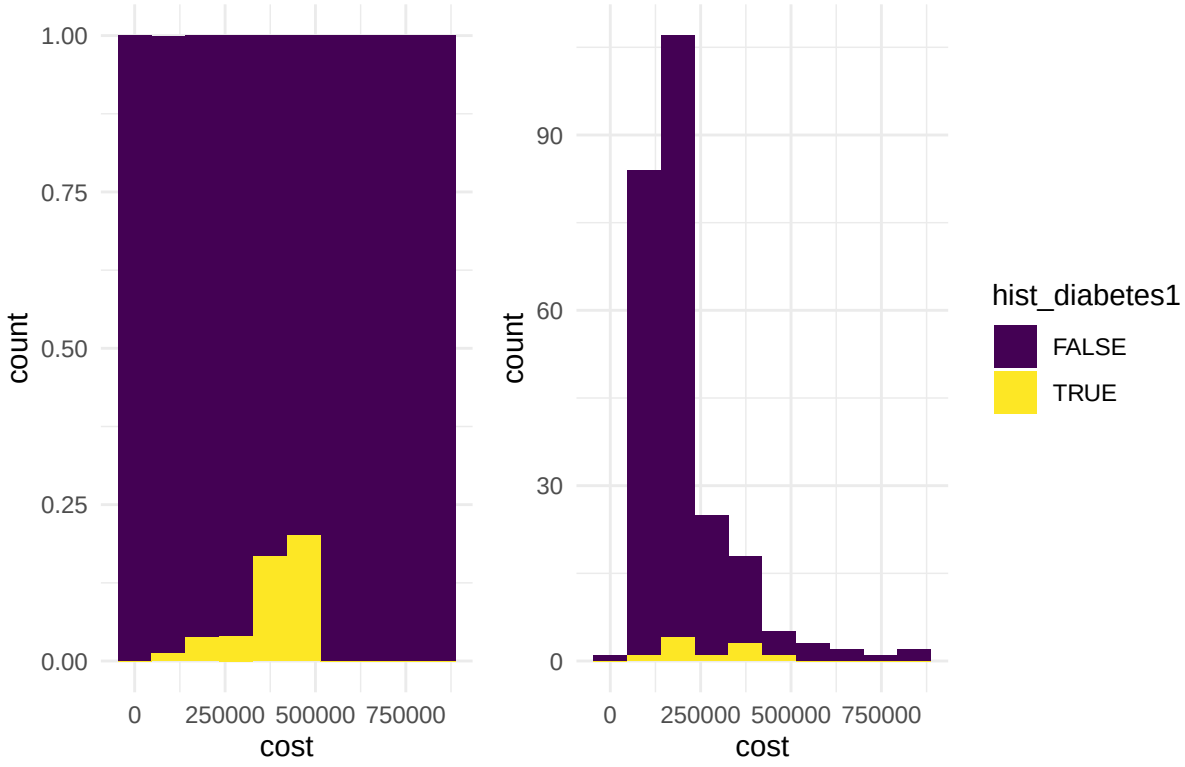


Figure 24: Plotting hist_diabetes2 against Target Variable

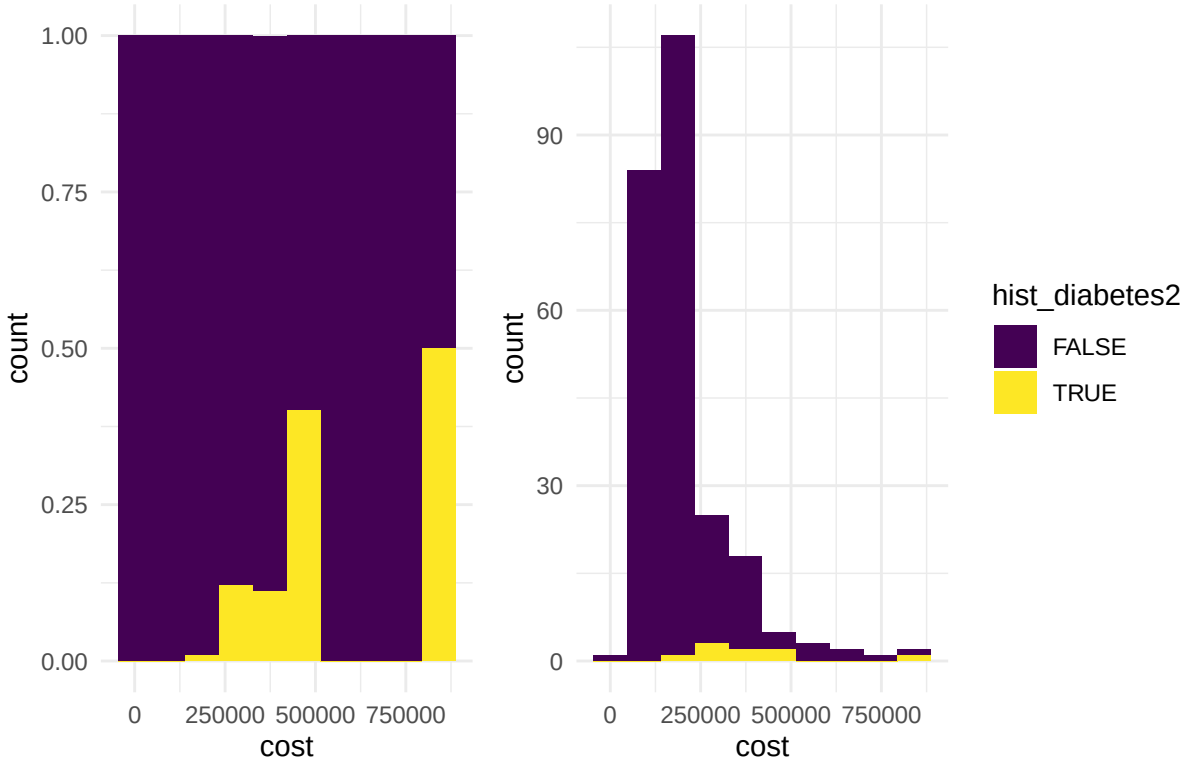


Figure 25: Plotting hist_hypertension1 against Target Variable

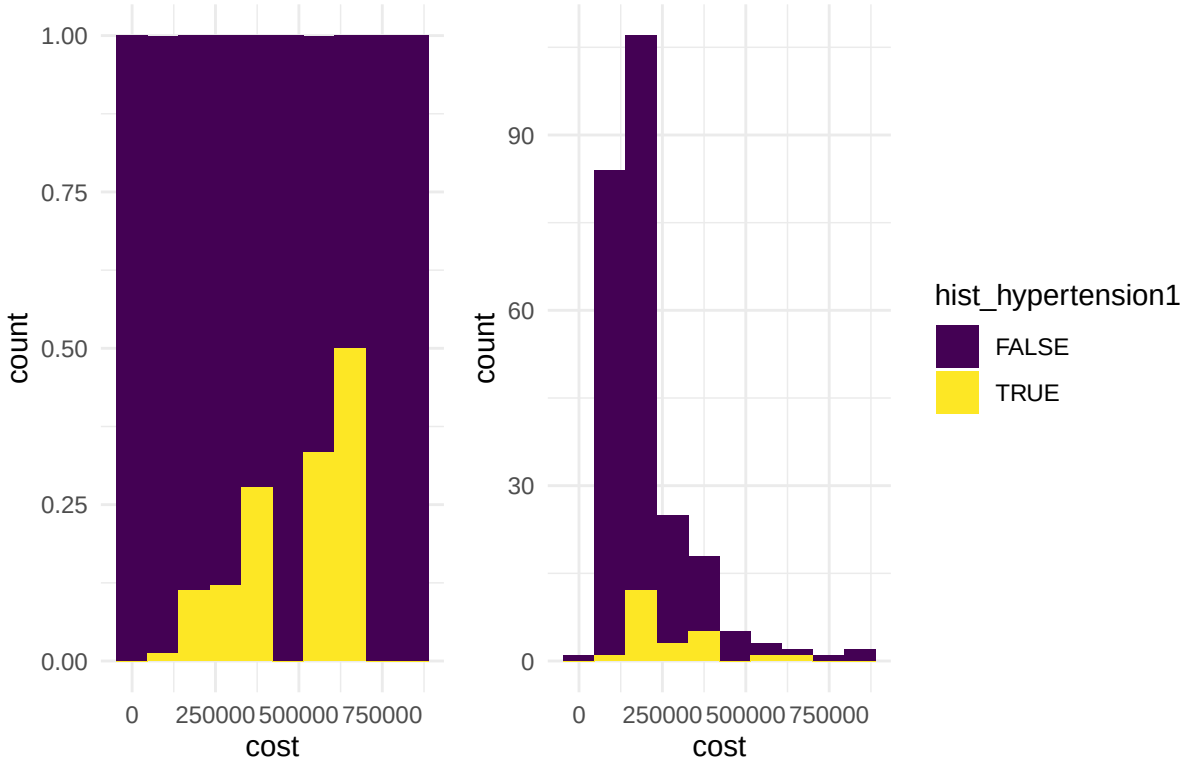


Figure 26: Plotting hist_hypertension2 against Target Variable

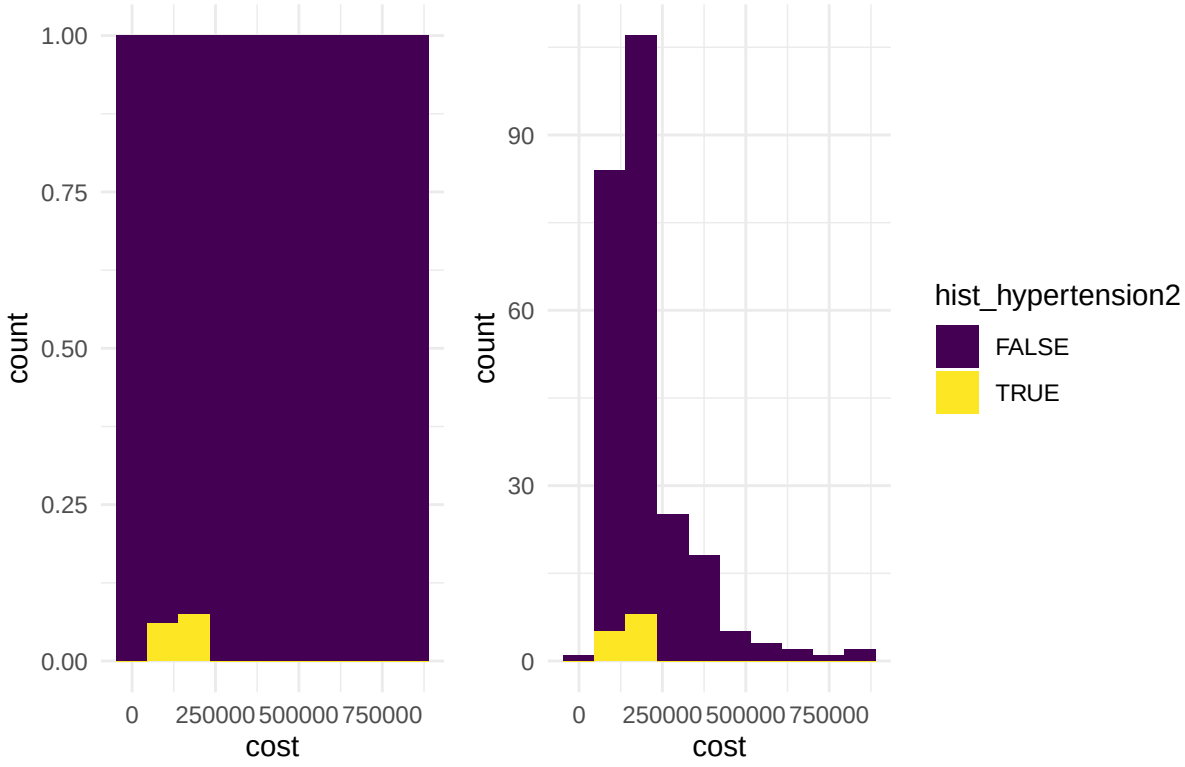


Figure 27: Plotting hist_hypertension3 against Target Variable

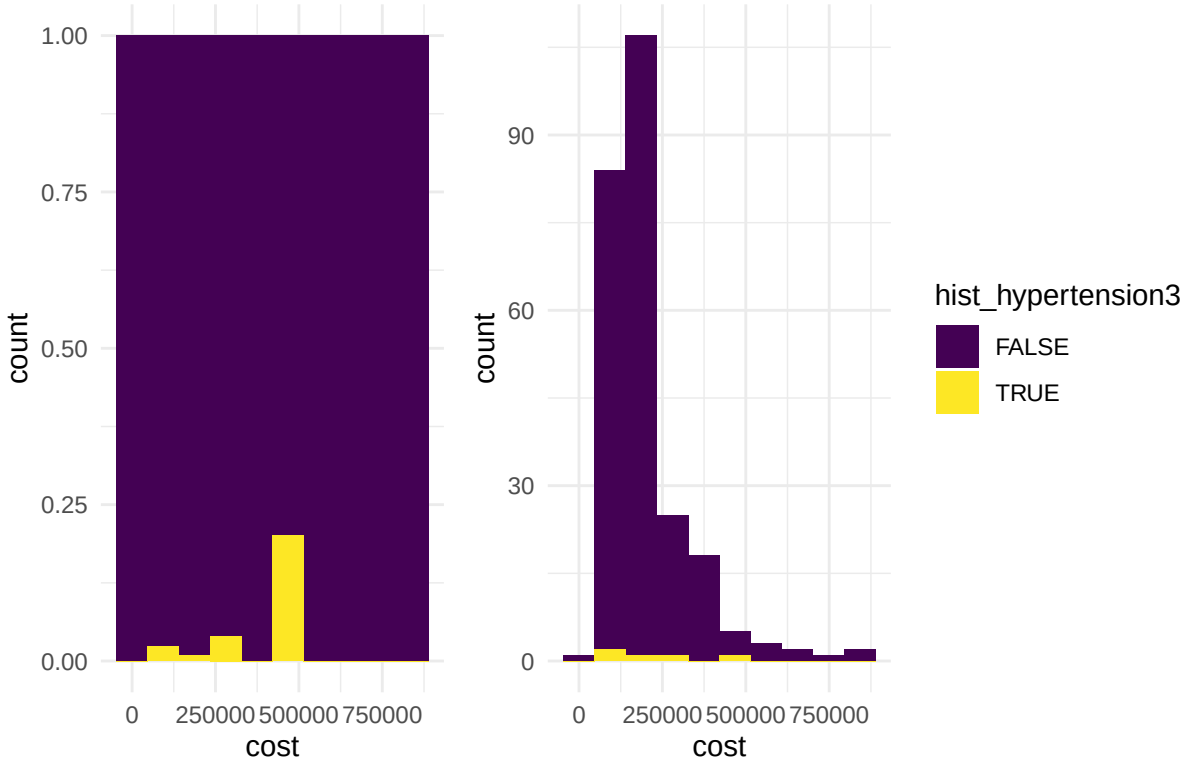
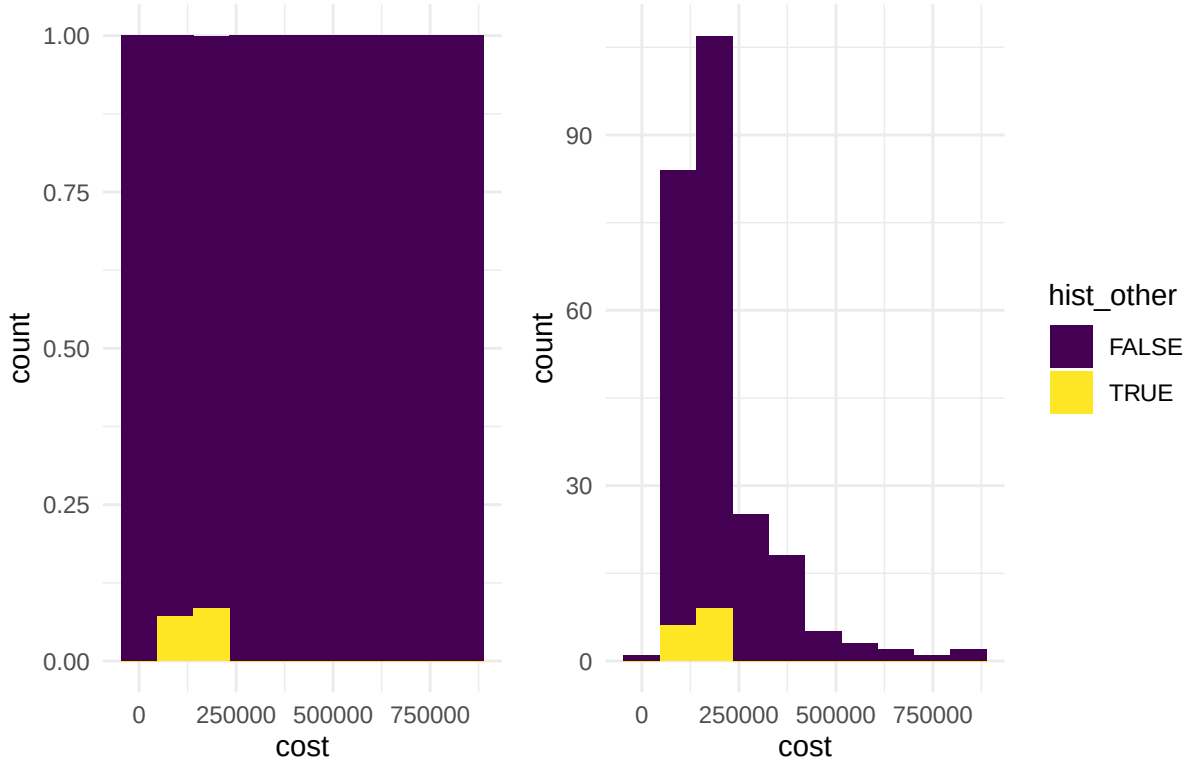


Figure 28: Plotting hist_other against Target Variable

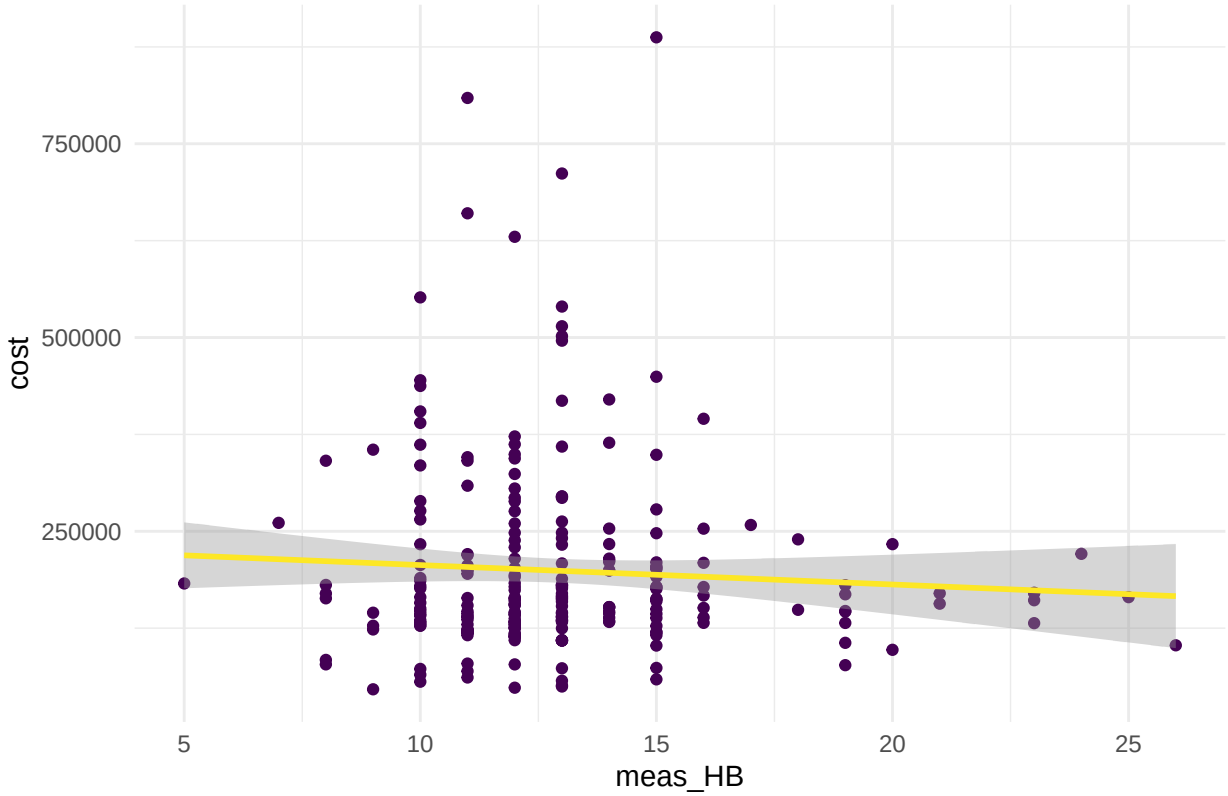



```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 2 rows containing non-finite values (stat_smooth).

## Warning: Removed 2 rows containing missing values (geom_point).
```

Figure 29: Plotting meas_HB against Target Variable

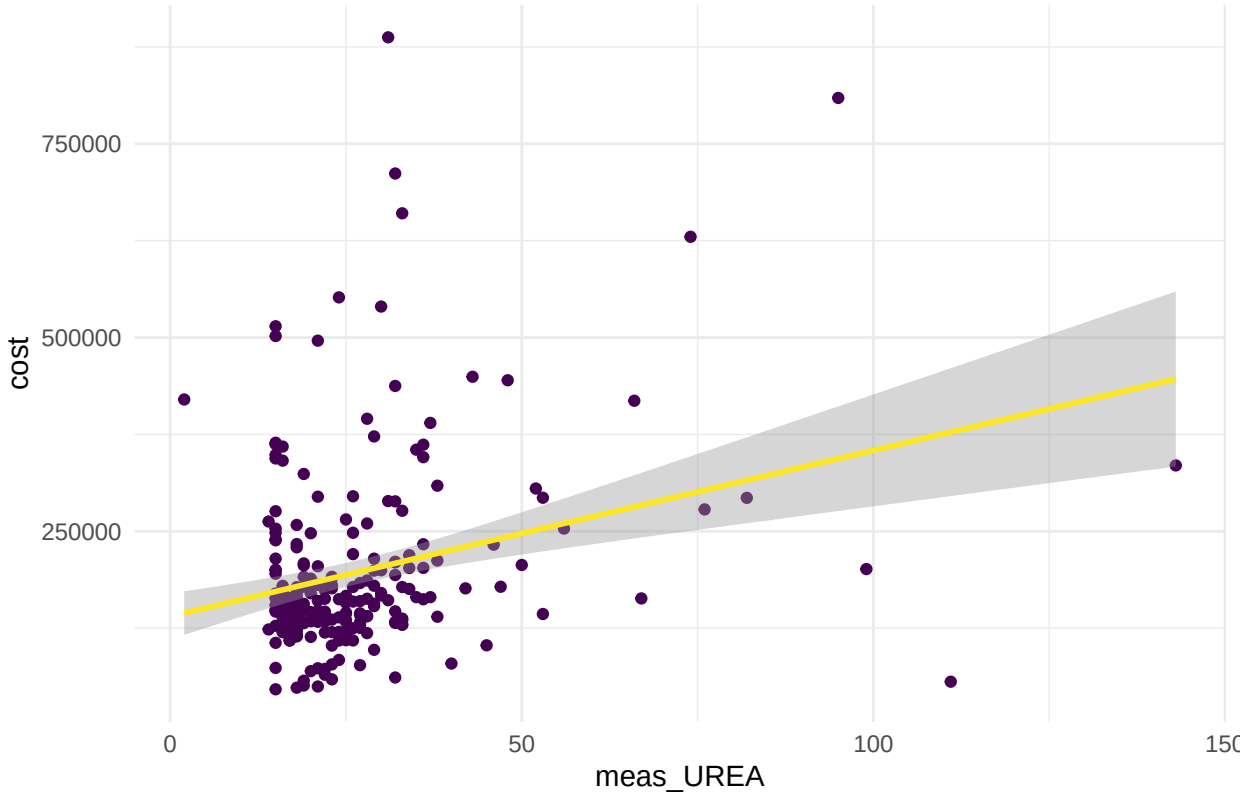



```
## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 13 rows containing non-finite values (stat_smooth).

## Warning: Removed 13 rows containing missing values (geom_point).
```

Figure 30: Plotting meas_UREA against Target Variable




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

## Warning: Removed 33 rows containing non-finite values (stat_smooth).

## Warning: Removed 33 rows containing missing values (geom_point).
```

Figure 31: Plotting meas_CREATININE against Target Variable

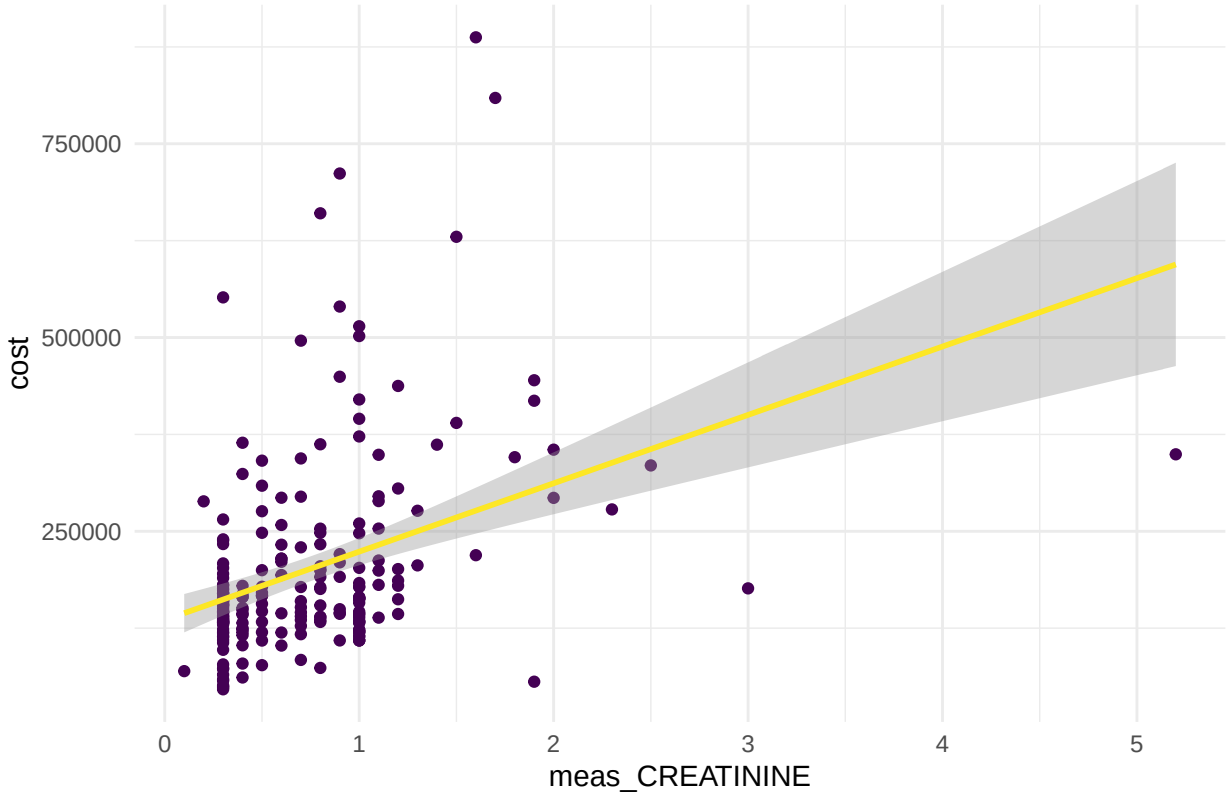


Figure 32: Plotting arr_walkin against Target Variable

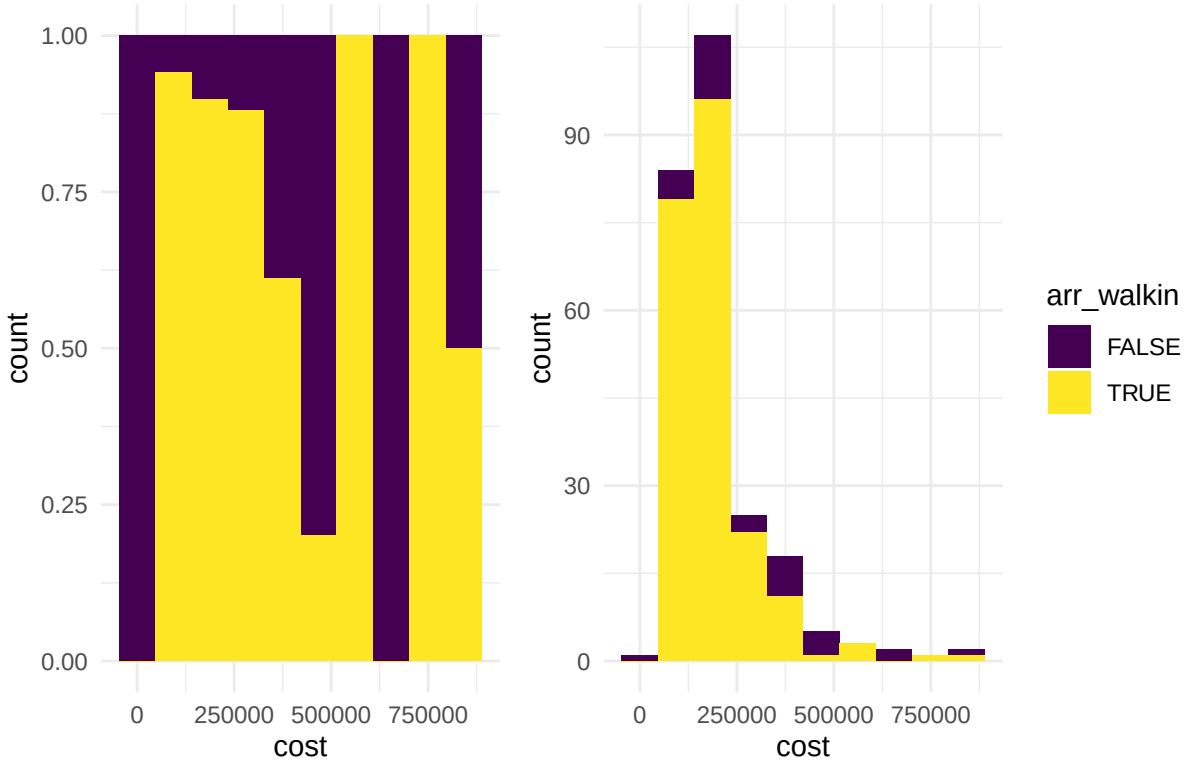


Figure 33: Plotting arr_ambulance against Target Variable

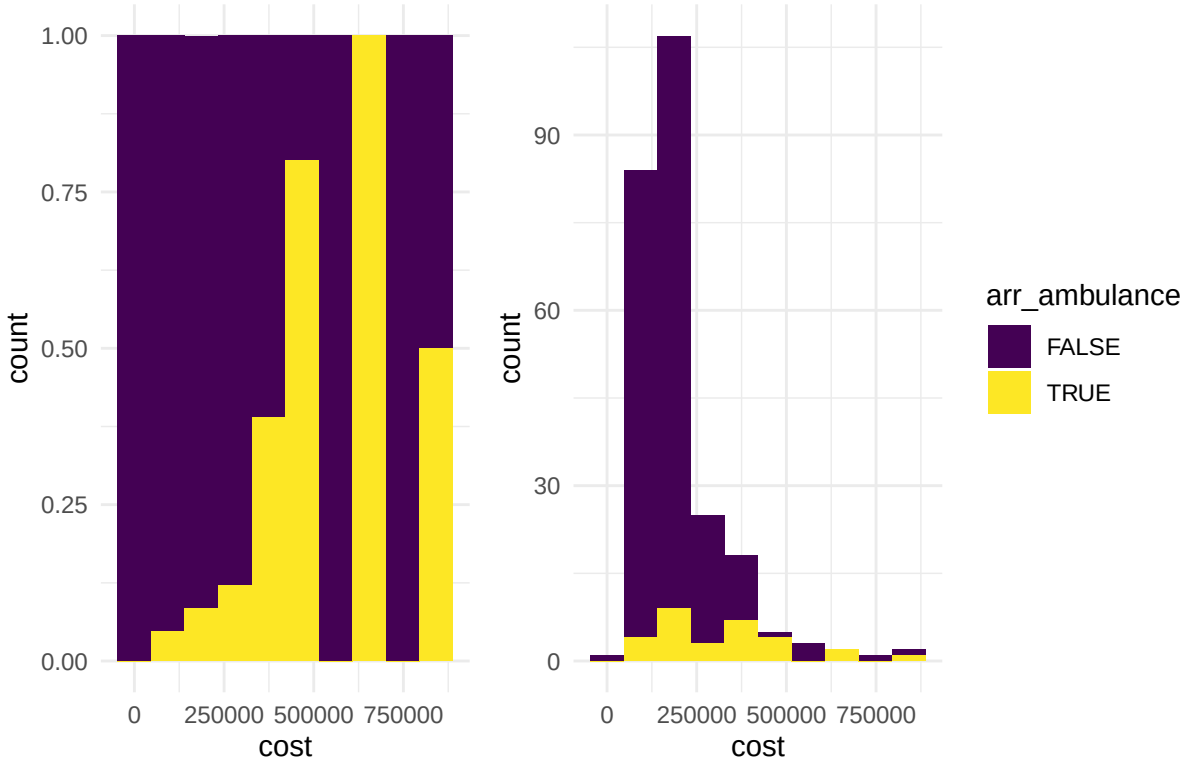


Figure 34: Plotting arr_transfer against Target Variable

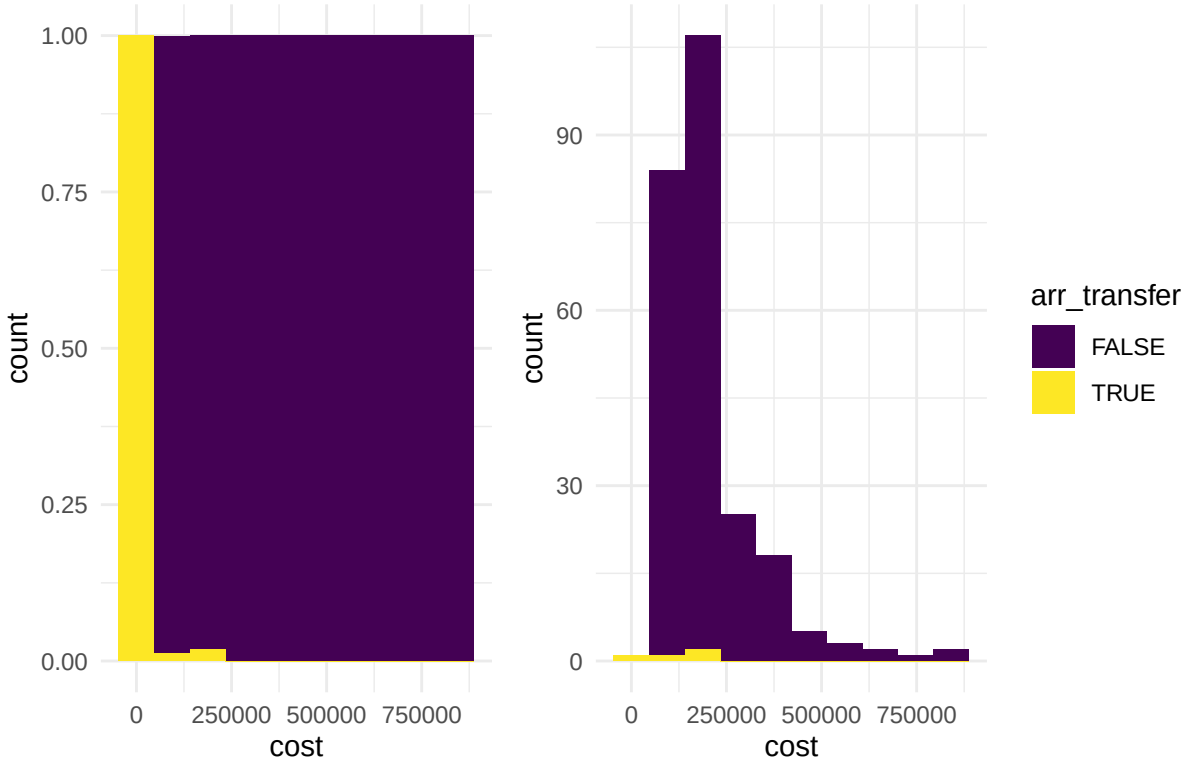


Figure 35: Plotting is_alert against Target Variable

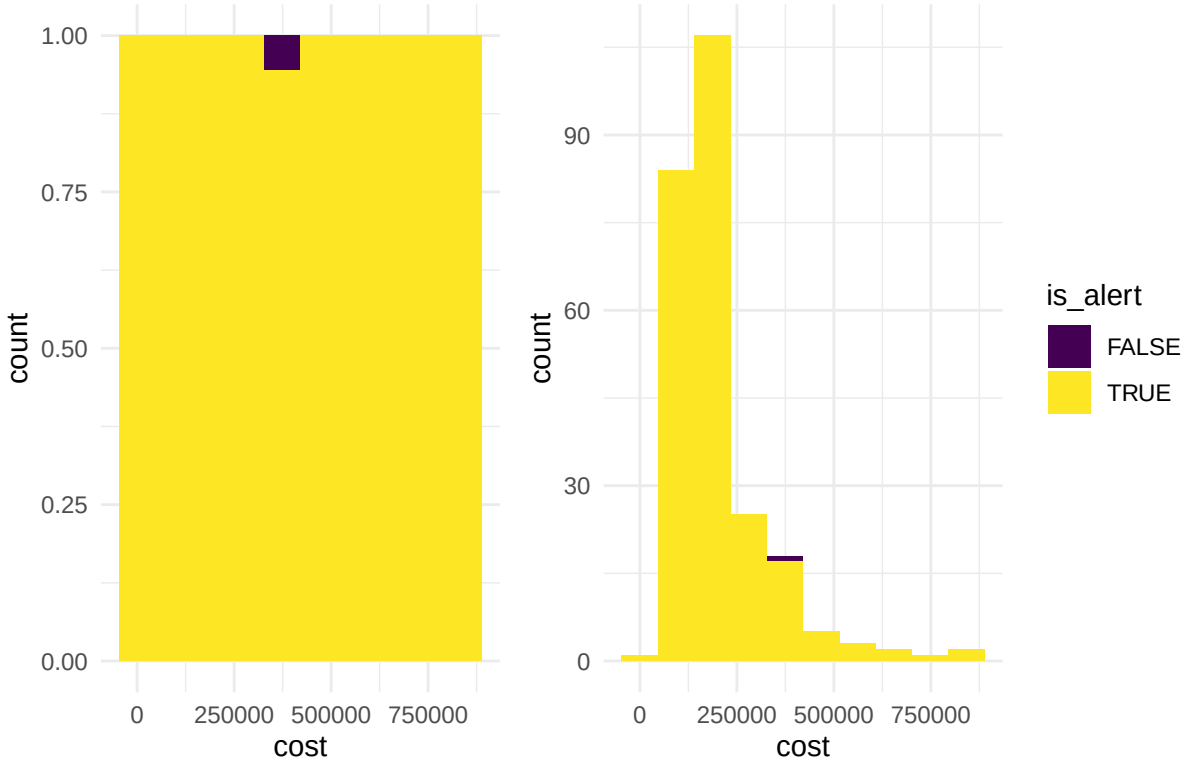
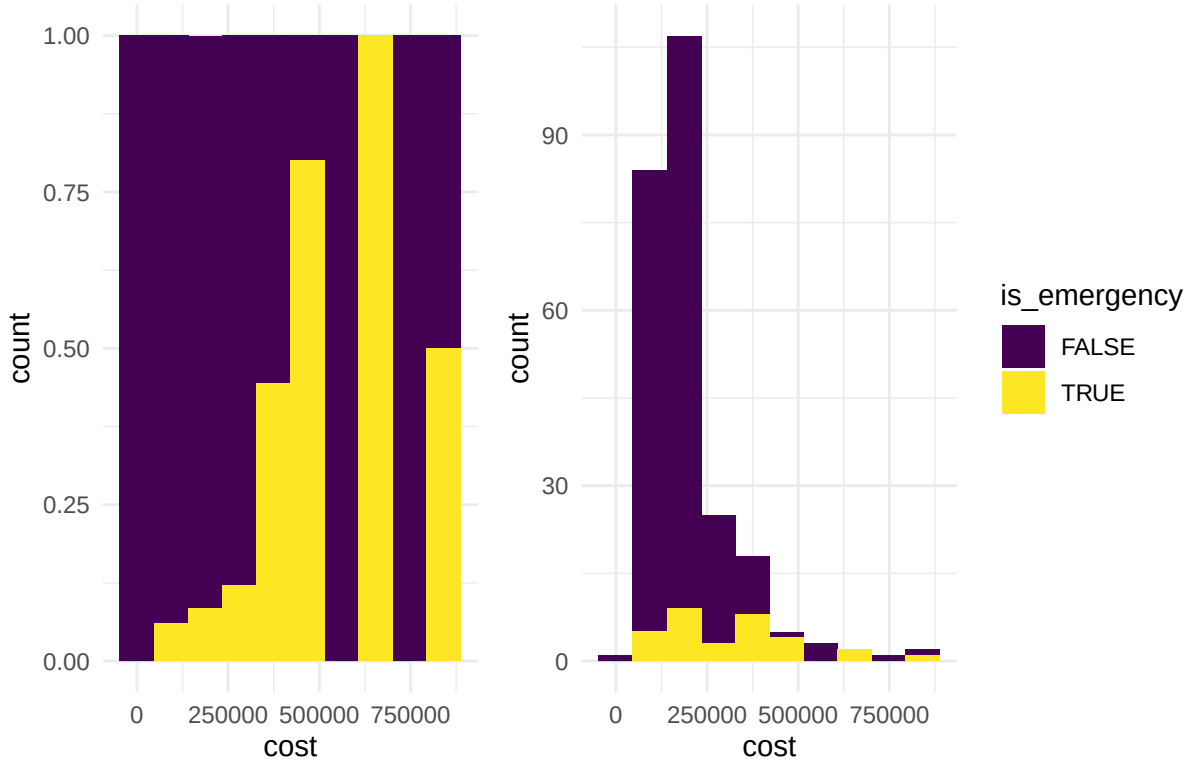
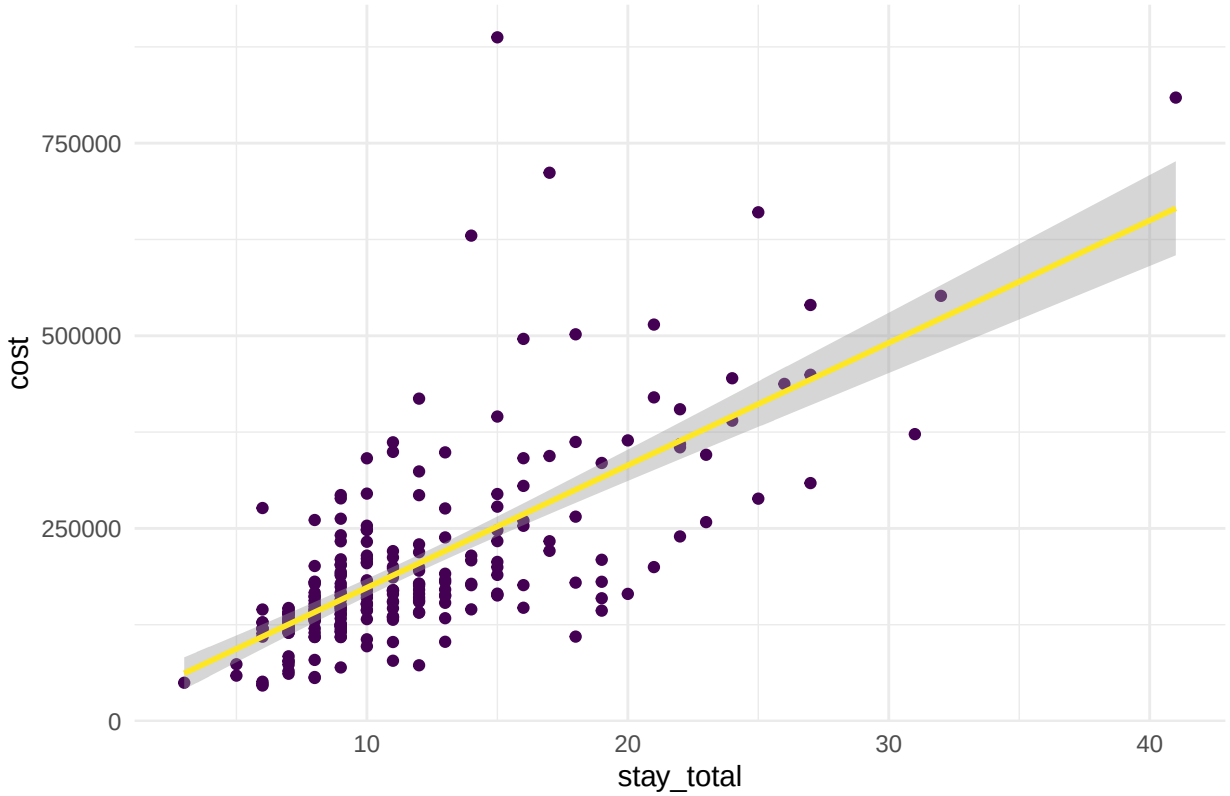


Figure 36: Plotting is_emergency against Target Variable



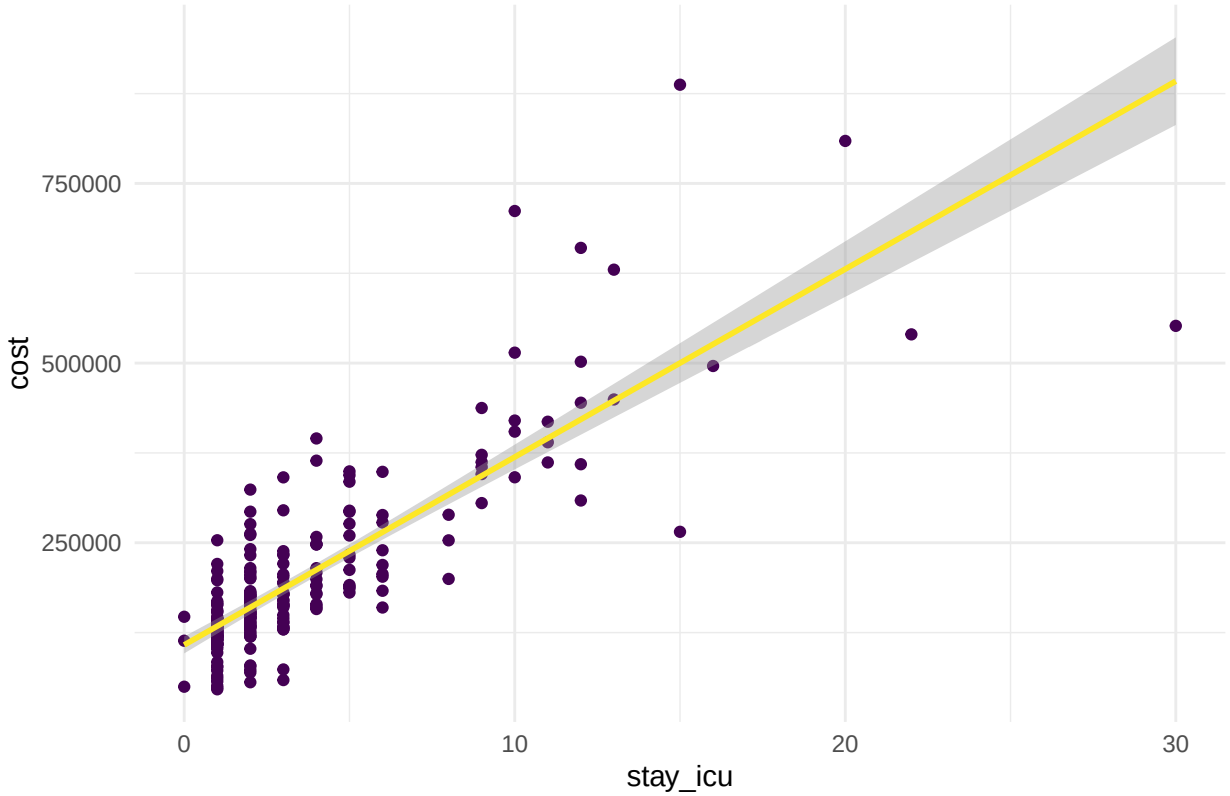

```
## `geom_smooth()` using formula 'y ~ x'
```

Figure 37: Plotting stay_total against Target Variable




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

Figure 38: Plotting stay_icu against Target Variable




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

Figure 39: Plotting stay_ward against Target Variable

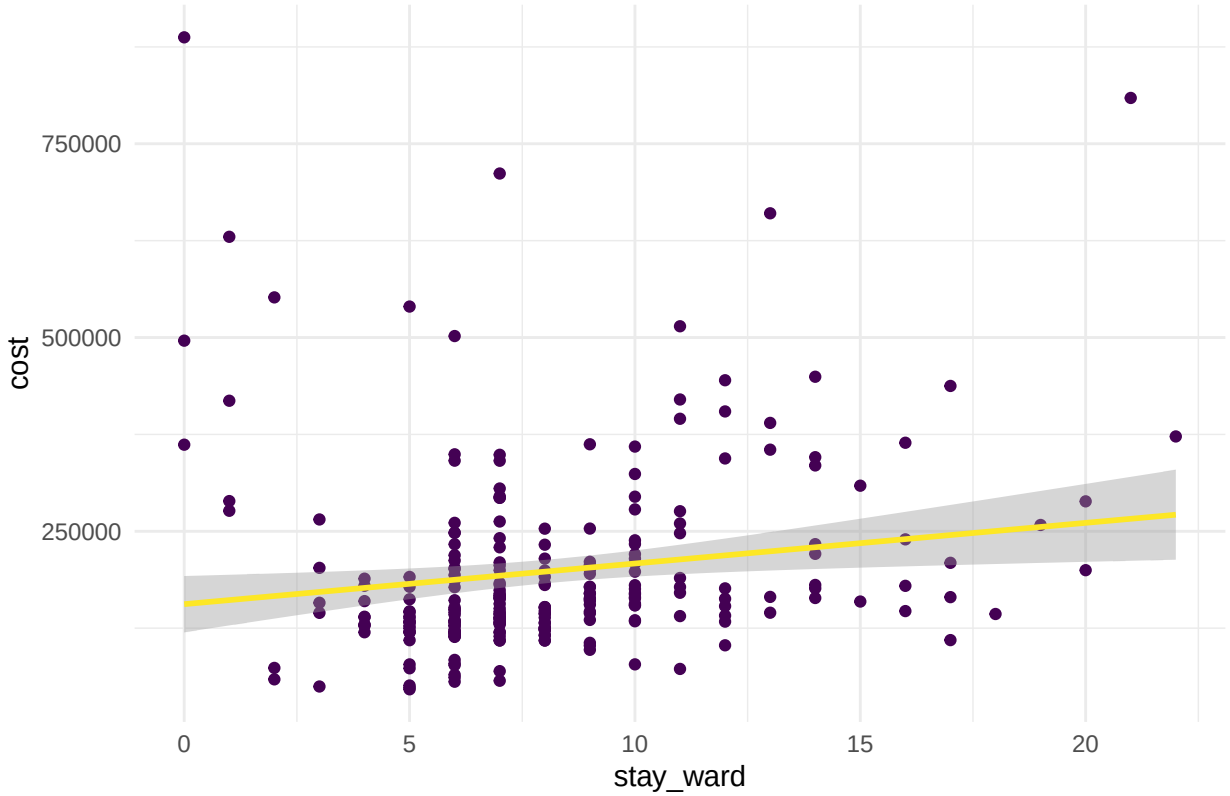
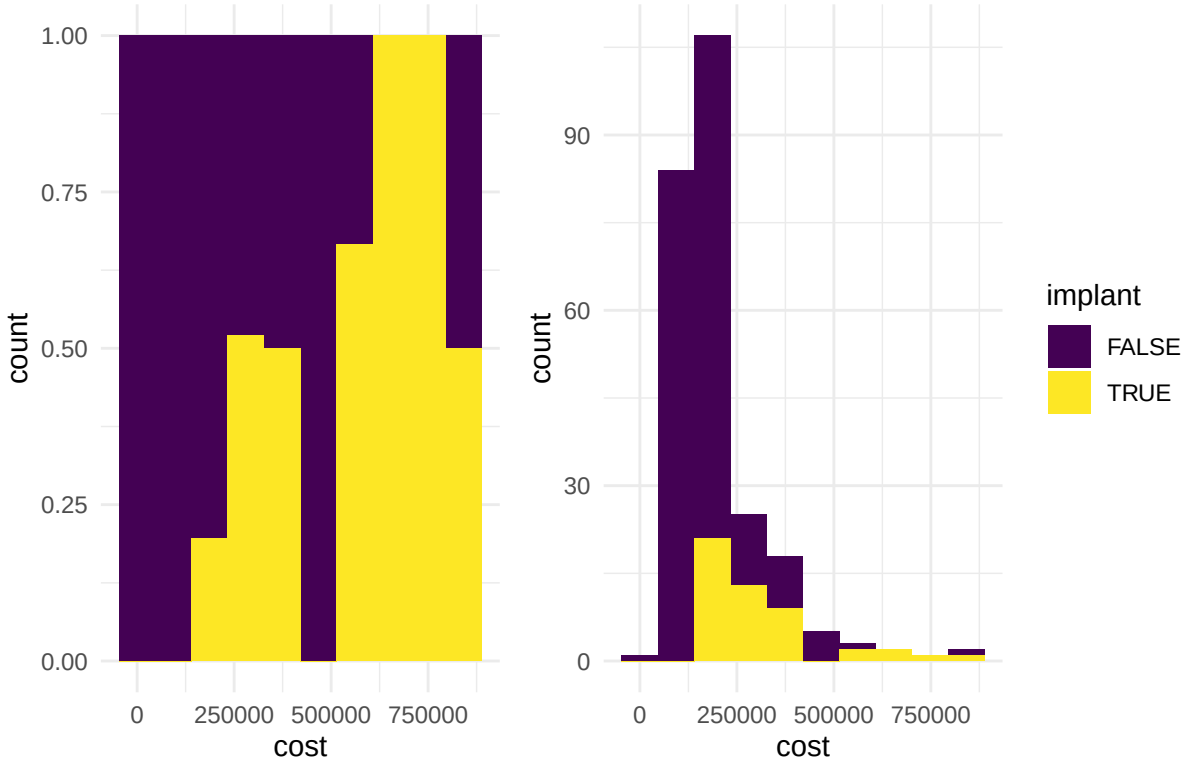
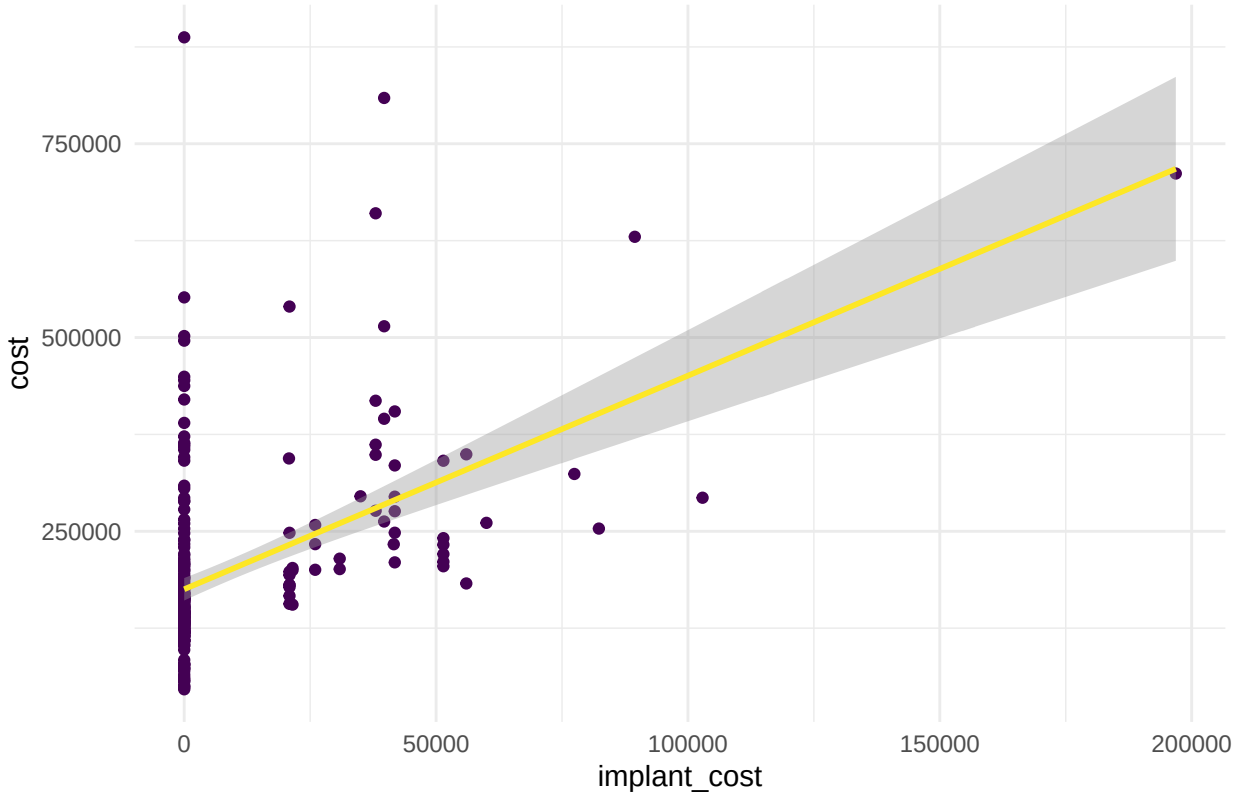


Figure 40: Plotting implant against Target Variable




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

Figure 41: Plotting implant_cost against Target Variable



Splitting Train/Test Sets

```
######################################## SPLITTING ########################################
d <- df %>% na.omit()
na_table(d)
```

```
##                    prop_NA
## id                       0
## demo_age                 0
## demo_female              0
## demo_unmarried           0
## code_ACHD                0
## code_CAD_DVD             0
## code_CAD_SVD             0
## code_CAD_TVD             0
## code_CAD_VSD             0
## code_OS_ASD              0
## code_PM_VSD              0
## code_RHD                 0
## code_othr_heart          0
## code_othr_respi          0
## code_othr_genrl          0
## code_othr_nerve          0
```

```
## code_othr_terta          0
## body_wgt                 0
## body_hgt                 0
## body_hrpulse             0
## body_bphigh              0
## body_bplow               0
## body_rr                  0
## hist_diabetes1           0
## hist_diabetes2           0
## hist_hypertension1       0
## hist_hypertension2       0
## hist_hypertension3       0
## hist_other               0
## meas_HB                  0
## meas_UREA                0
## meas_CREATININE          0
## arr_walkin               0
## arr_ambulance            0
## arr_transfer             0
## is_alert                 0
## is_emergency             0
## cost                     0
## cost_ln                  0
## stay_total               0
## stay_icu                 0
## stay_ward                0
## implant                  0
## implant_cost             0
```

```
set.seed(8675309)
splitkey <- initial_split(d,
                          prop = .9,
                          strata = cost)
d.train <- training(splitkey)
d.test  <- testing(splitkey)
```

Single Regression: Body Weight

```
######################################## SINGLE-X REGRESSION ########################################
# Set preprocessing recipe
shortLM.rc <- recipe(cost ~ body_wgt,
                     data = d.train)

# Set engine
shortLM.en <- linear_reg() %>% set_engine("lm")

# Combining recipe + engine
shortLM.wf <- workflow() %>%
  add_model(shortLM.en) %>%
  add_recipe(shortLM.rc)
```

```
# Fitting
set.seed(8675309)
shortLM.ft <- shortLM.wf %>% fit(data=d.train)

# Fit Summary
shortLM.sm <- shortLM.ft %>%
  extract_fit_parsnip() %>%
  tidy()

shortLM.sm
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic      p.value
##   <chr>          <dbl>     <dbl>     <dbl>        <dbl>
## 1 (Intercept)  106635.    19541.       5.46 0.000000170
## 2 body_wgt       2404.      416.       5.78 0.0000000348
```

```
######################################## TEST SET ACCURACY: MAPE ########################################
shortLM.pred <- shortLM.ft %>% last_fit(splitkey) %>% collect_predictions()

mape(shortLM.pred,
     truth = cost,
     estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mape    standard        50.8
```

Multiple Regression: Kitchen Sink without Leaks

```
######################################## MULTIPLE REGRESSION ########################################
# Set preprocessing recipe
multiLM.rc <- recipe(cost ~ .,
                     data = d.train) %>%
  step_mutate(body_bmi  = body_wgt/((body_hgt/100)^2), role = "redundant") %>%
  step_mutate(wgt_under = ifelse(body_bmi < 18.5,                TRUE,FALSE),
              wgt_norml = ifelse(18.5 <= body_bmi & body_bmi < 25,TRUE,FALSE),
              wgt_overw = ifelse(25 <= body_bmi & body_bmi <= 30, TRUE,FALSE),
              wgt_obese = ifelse(30 < body_bmi,                  TRUE,FALSE)) %>%
  update_role(c(cost_ln,stay_total,implant_cost,body_wgt,body_hgt),new_role = "redundant") %>%
  update_role(c(
    # Setting leakages
    tidyselect::starts_with("code"),
    tidyselect::starts_with("stay"),
    tidyselect::starts_with("implant")
  ),new_role = "leakage") %>%
  update_role(id,new_role = "ID") %>%
  step_zv(all_numeric_predictors())
```

```r
# Set engine
multiLM.en <- linear_reg() %>% set_engine("lm")

# Combining recipe + engine
multiLM.wf <- workflow() %>%
  add_model(multiLM.en) %>%
  add_recipe(multiLM.rc)

# Fitting
set.seed(8675309)
multiLM.ft <- multiLM.wf %>% fit(data=d.train)

# Fit Summary
multiLM.sm <- multiLM.ft %>%
  extract_fit_parsnip() %>%
  tidy()

multiLM.sm %>%
  #filter(term != "(Intercept)") %>%
  arrange(p.value)
```

```
## # A tibble: 26 x 5
##    term                    estimate std.error statistic p.value
##    <chr>                      <dbl>     <dbl>     <dbl>   <dbl>
##  1 demo_age                   2337.      869.     2.69  0.00797
##  2 body_hrpulse               1258.      567.     2.22  0.0281
##  3 hist_diabetes2TRUE        98639.    45519.     2.17  0.0318
##  4 meas_CREATININE           61224.    34452.     1.78  0.0776
##  5 body_bphigh               -1164.      694.    -1.68  0.0958
##  6 arr_walkinTRUE            87433.    64116.     1.36  0.175
##  7 hist_otherTRUE           -49817.    36954.    -1.35  0.180
##  8 hist_hypertension2TRUE   -54515.    44894.    -1.21  0.227
##  9 body_rr                    3068.     2634.     1.16  0.246
## 10 wgt_underTRUE            -36983.    37263.    -0.992 0.323
## # ... with 16 more rows
```

```r
######################################### TEST SET ACCURACY: MAPE ########################################
multiLM.pred <- multiLM.ft %>% last_fit(splitkey) %>% collect_predictions()
```

```
## ! train/test split: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-defici...
```

```r
mape(multiLM.pred,
     truth = cost,
     estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mape    standard        36.2
```