```
#Libraries
library(broom)
library(car)
library(caret)
library(corrplot)
library(cowplot)
library(dplyr)
library(e1071)
library(Epi)
library(flextable)
library(forcats)
library(ggplot2)
library(glmnet)
library(grid)
library(gtsummary)
library(haven)
library(janitor)
library(MASS)
library(naniar)
library(officer)
library(patchwork)
library(pROC)
library(randomForest)
library(rmi)
library(themis)
library(tidyverse)
library(tools)
library(vcd)
library(vip)
library(VSURF)

#-------------------------------------------------------------------------------
#Uploading dataset
dw.raw <- read_sav('overlim15data.sav')
dw.v0 <- dw.raw[,-1]
names(dw.v0)

#~~~~~~~~~Export outputs to word~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
toword_fx <- function(tbl, filename) {
  doc <- read_docx()
  doc <- doc %>% body_add_flextable(value = tbl)
  print(doc, target = filename)}

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Converting missing data to NA
dw.raw[dw.raw < 0] <- NA

#Labells to factors

sapply(dw.v0, function(x) n_distinct(x))

dw.v0 <- dw.v0 %>% mutate_if(is.labelled, as_factor)
str(dw.v0)

# Convert to numeric
convert_factors2numeric <- function(df, columns) {
  df[, columns] <- lapply(df[, columns], function(x) {
      if (is.factor(x)) {
        as.numeric(as.character(x))} else {
        x}})
    return(df)}
str(dw.v0)

# Function: Checking factors
factor_fx <- function(data) {
  factor_levels <- sapply(data, function(x) {
    if (is.factor(x)) {levels(x)}
    else if (is.numeric(x)) {'numeric variable'}
    else {NULL}})
```

```
  return(factor_levels)}

factor_fx(dw.v0)

# Variable names to lowercase
names(dw.v0) <- tolower(names(dw.v0))
names(dw.v0)

# Function to clean factor levels
factornames_fx <- function(dw.v0) {
  for (col in names(dw.v0)) {
    if (is.factor(dw.v0[[col]])) {
      levels(dw.v0[[col]]) <- tolower(gsub("[: -]", "_", levels(dw.v0[[col]])))}}
  return(dw.v0)}

dw.v0 <- factornames_fx(dw.v0)

factor_fx(dw.v0)

# Changing order of factors: lifesat2
dw.v0$lifesat2 <- factor(dw.v0$lifesat2,
                      levels = c('above_mode',
                                 'mode',
                                 'below_mode'))

levels(dw.v0$lifesat2)
summary(dw.v0$lifesat2)

# Changing order of factors: limitac_h
dw.v0$limitac_h <- factor(dw.v0$limitac_h,
                         levels = c('no_li',
                                    'not_at_all',
                                    'a_little',
                                    'a_lot'))

levels(dw.v0$limitac_h)
summary(dw.v0$limitac_h)

# Changing order of factors: cig
dw.v0$cig <- factor(dw.v0$cig,
                         levels = c('never_smoked', 'ex_smoker', 'light_smokers',
                                    'moderate_smokers', 'heavy_smokers'))

levels(dw.v0$cig)
summary(dw.v0$cig)

factor_fx(dw.v0)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Creating urban subset
dwu.v0 <- dw.v0 %>% filter(urbrur_all == 'urban')%>% dplyr::select(-urbrur_all)
dim(dwu.v0)

# Creating rural subset
dwr.v0 <- dw.v0 %>% filter(urbrur_all == 'rural')%>% dplyr::select(-urbrur_all)
dim(dwr.v0)

#~~~~~~~~~~~~~~~~Descriptive Statistics ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Descriptive statistics: rural vs urban (urbrur_all)
dw.v0_stxurbrur <- dw.v0 %>%
  tbl_summary(by = urbrur_all,
              type = all_continuous() ~ 'continuous2',
              statistic = list(all_continuous() ~ c('{mean} (sd={sd})',
                                                    '{median} ({p25}, {p75})',
                                                    '{min} - {max}'),
                               all_categorical() ~ '{n} ({p}%)'),
              digits = c(all_categorical() ~ c(0, 2),
                         all_continuous() ~ 2) ,
              missing = 'no') %>%
```

```
  add_n() %>%
  modify_header(label ~ '**Variable**') %>%
  modify_spanning_header(c('stat_1', 'stat_2') ~ '**Population Classification**') %>%
  bold_labels()
dw.v0_stxurbrur.flex <- as_flex_table(dw.v0_stxurbrur)
dw.v0_stxurbrur.flex
toword_fx(dw.v0_stxurbrur.flex, 'dw0_xurbrur.docx')

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Descriptive statistics: overlim15
# Overall dataset
dw.v0_stxover <- dw.v0 %>%
  tbl_summary(by = overlim15, percent = 'row',
              type = all_continuous() ~ 'continuous2',
              statistic = list(all_continuous() ~ c('{mean} (sd={sd})',
                                                    '{median} ({p25}, {p75})',
                                                    '{min} - {max}'),
                               all_categorical() ~ '{n} ({p}%)'),
              digits = c(all_categorical() ~ c(0, 2),
                         all_continuous() ~ 2) ,
              missing = 'no') %>%
  modify_header(label ~ '**Variable**') %>%
  add_p(pvalue_fun = ~ style_pvalue(.x, digits = 2)) %>%
  modify_spanning_header(c('stat_1', 'stat_2') ~ '**Above the limit**') %>%
  bold_labels()
dw.v0_stxover.flex <- as_flex_table(dw.v0_stxover)
dw.v0_stxover.flex
toword_fx(dw.v0_stxover.flex, 'dw0_stxover.docx')

# Urban dataset
dwu.v0_stxover <- dwu.v0 %>%
  tbl_summary(by = overlim15, percent = 'row',
              type = all_continuous() ~ 'continuous2',
              statistic = list(all_continuous() ~ c('{mean} (sd={sd})',
                                                    '{median} ({p25}, {p75})',
                                                    '{min} - {max}'),
                               all_categorical() ~ '{n} ({p}%)'),
              digits = c(all_categorical() ~ c(0, 2),
                         all_continuous() ~ 2) ,
              missing = 'no') %>%
  modify_header(label ~ '**Variable**') %>%
  add_p(pvalue_fun = ~ style_pvalue(.x, digits = 2)) %>%
  modify_spanning_header(c('stat_1', 'stat_2') ~ '**Above the limit**') %>%
  bold_labels()
dwu.v0_stxover.flex <- as_flex_table(dwu.v0_stxover)
dwu.v0_stxover.flex
toword_fx(dwu.v0_stxover.flex, 'dwu_stxover.docx')

# Rural dataset
dwr.v0_stxover <- dwr.v0 %>%
  tbl_summary(by = overlim15, percent = 'row',
              type = all_continuous() ~ 'continuous2',
              statistic = list(all_continuous() ~ c('{mean} (sd={sd})',
                                                    '{median} ({p25}, {p75})',
                                                    '{min} - {max}'),
                               all_categorical() ~ '{n} ({p}%)'),
              digits = c(all_categorical() ~ c(0, 2),
                         all_continuous() ~ 2) ,
              missing = 'no') %>%
  modify_header(label ~ '**Variable**') %>%
  add_p(pvalue_fun = ~ style_pvalue(.x, digits = 2)) %>%
  modify_spanning_header(c('stat_1', 'stat_2') ~ '**Above the limit**') %>%
  bold_labels()
dwr.v0_stxover.flex <- as_flex_table(dwr.v0_stxover)
dwr.v0_stxover.flex
toword_fx(dwr.v0_stxover.flex, 'dwr_stxover.docx')


#~~~~~~~~~~~~~Visualisation x urban-rural~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
fx_plot.cat <- function(data, var, varby, var_des, cat_labels = NULL) {
  filtered_data <- data %>%
    filter(!is.na(.data[[var]]), !is.na(.data[[varby]])) %>%
    count(.data[[var]], .data[[varby]]) %>%
    group_by(.data[[varby]]) %>%
    mutate(total_count = sum(n),
           percent = n / total_count * 100) %>% ungroup()

  unfiltered_data <- data %>%
    filter(!is.na(.data[[var]])) %>%
    count(.data[[var]]) %>%
    mutate(!!varby := "All",
           total_count = sum(n),
           percent = n / total_count * 100)

  combined_data <- bind_rows(unfiltered_data, filtered_data) %>%
    mutate(interaction_label = interaction(.data[[varby]], .data[[var]]))

  combined_data[[varby]] <- factor(combined_data[[varby]], levels = c("All", "urban",
"rural"))

  custom_labels <- function(x) {
    if (!is.null(cat_labels) && x %in% names(cat_labels)) {
      return(cat_labels[[x]])}
    x <- as.character(x)
    x <- gsub('_', ' ', x)
    x <- toTitleCase(x)
    return(x)}

  ggplot(combined_data, aes(x = .data[[var]], y = percent, fill = .data[[varby]])) +
    geom_bar(stat = 'identity', position = 'dodge', color = 'black') +
    scale_y_continuous(limits = c(0, 100)) +
    labs(x = '', y = 'Percentage', fill = 'Categories') +
    scale_fill_brewer(palette = 'Paired') +
    scale_x_discrete(labels = function(x) sapply(x, custom_labels)) +
    theme_minimal() +
    theme(legend.position = 'none',
          plot.title = element_text(size = 12, face = 'bold'),
          axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.text.x = element_text(size = 11, face = 'bold',
                                     angle = 45, hjust = 1, vjust = 1)) +
    ggtitle(sprintf('%s (%s)', var_des, var))}

#~~~~~~~~~~~~~~~~
# Sociodemographic variables
print(unique(dw.v0$ag16g10))
cat_labels <- c('25_34' = '25 to 34',
                '35_44' = '35 to 44',
                '45_54' = '45 to 54',
                '55_64' = '55 to 64',
                'rest_UK' = 'Rest of the UK',
                'white_scottish' = 'White: Scottish',
                'white_restUK' = 'White: Other British',
                'white_other' = 'White: Other',
                'other_minority'= 'Other minority ethnic',
                'church_scotland' = 'Church of Scotland',
                'married_partner' = 'Married or civil partnership',
                'as_married' = 'Living as married',
                'divorced_dissolved' = 'Divorced or dissolved',
                'widowed' = 'Widowed or surviving partner')

p1 <- fx_plot.cat(dw.v0, 'ag16g10', 'urbrur_all',
                  'Age 16+ in 10 year bands', cat_labels)  +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
p2 <- fx_plot.cat(dw.v0, 'sex', 'urbrur_all',
                  'Sex of respondent')
p3 <- fx_plot.cat(dw.v0, 'birthpla3', 'urbrur_all',
```

```
                  'Country of birth', cat_labels)
p4 <- fx_plot.cat(dw.v0, 'ethnic05', 'urbrur_all',
                  'Ethnic background', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
p5 <- fx_plot.cat(dw.v0, 'religi04', 'urbrur_all',
                  'Religion', cat_labels)
p6 <- fx_plot.cat(dw.v0, 'maritalg', 'urbrur_all',
                  'Marital status', cat_labels) +
  theme(legend.position = 'bottom',
        plot.margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm"))


sociodemo_plots <- p1 + p2 + p3 + p4 + p5 + p6 +
  plot_layout(ncol = 3, widths = c(2, 2, 2),
              heights = unit(c(3.5, 3.5), 'cm')) +
  theme(plot.margin=unit(c(0.1, 0.1, 0.1, 0.1), "cm"))

print(sociodemo_plots)

#~~~~~~~~~~~~~~~~~
# Health variables
p7 <- fx_plot.cat(dw.v0, 'genhelf', 'urbrur_all',
                   'Self-assessed general health', cat_labels)  +
  theme(axis.title.y = element_text(size = 11, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 11, face = 'bold'))
p8 <- fx_plot.cat(dw.v0, 'limitac_h', 'urbrur_all',
                   'Whether any LTC limits activities', cat_labels)
p9 <- fx_plot.cat(dw.v0, 'lifesat2', 'urbrur_all',
                   'Life satisfaction', cat_labels)
p10 <- fx_plot.cat(dw.v0, 'adt10gptw', 'urbrur_all',
                   'Activity level', cat_labels) +
  theme(axis.title.y = element_text(size = 11, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 11, face = 'bold'))
p11 <- fx_plot.cat(dw.v0, 'cig', 'urbrur_all',
                   'Smoking', cat_labels)

layout <- 'AABBCC
DDDEEE'

health_plots <- p7 + p8 + p9 + p10 + p11 +
  plot_layout(ncol = 3, widths = c(2, 2, 2),
              heights = unit(c(3.5, 3.5), 'null'),
              design = layout) +
  theme(legend.position = 'bottom')
print(health_plots)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Economic situation variables
cat_labels <- c('hnc_d' = 'HNC/D',
                'professional' = 'I Professional',
                'managerial_technical' = 'II Managerial technical',
                'skilled_non_manual' = 'IIIN Skilled non-manual',
                'skilled_manual' = 'IIIM Skilled manual',
                'semiskilled_manual' = 'IV Semi-skilled manual',
                'unskilled_manual' = 'V Unskilled manual',
                'ilo_unemployed' = 'ILO Unemployed',
                'least_deprived' = '5th',
                'most_deprived' = '1st')

p12 <- fx_plot.cat(dw.v0, 'hedqul08', 'urbrur_all',
                  'Highest educational qualification', cat_labels)  +
  theme(axis.title.y = element_text(size = 11, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 11, face = 'bold'))
p13 <- fx_plot.cat(dw.v0, 'schrpg7', 'urbrur_all',
                  'Social Class of HRP', cat_labels)
p14 <- fx_plot.cat(dw.v0, 'neconacb', 'urbrur_all',
                  'Economic activity', cat_labels) +
  theme(axis.title.y = element_text(size = 11, angle = 90, face = 'bold'),
```

```
        axis.text.y = element_text(size = 11, face = 'bold'))
p15 <- fx_plot.cat(dw.v0, 'simd20_rpa', 'urbrur_all',
                  'SIMD 2020 quintiles', cat_labels) +
  theme(legend.position = 'bottom')

econ_plots <- p12 + p13 + p14 + p15 +
  plot_layout(ncol = 2, widths = c(3, 3),
              heights = unit(c(4, 4), 'cm')) +
  theme(plot.margin=unit(c(0.1, 0.1, 0.1, 0.1), "cm"))
print(econ_plots)

#~~~~~~~~~~~~~~~~~~
# Consumption EtOH
fx_plot.numoh <- function(data, num_var, cat_var, var_des) {
  data_for_plot <- data %>%
    filter(!is.na(.data[[num_var]]), !is.na(.data[[cat_var]]))

  data_for_plot[[num_var]] <- ifelse(data_for_plot[[num_var]] == 0, 0.001,
data_for_plot[[num_var]])

  overall_data <- data_for_plot %>%
    mutate(!!cat_var := "All")

  combined_data <- bind_rows(data_for_plot, overall_data)

  combined_data[[cat_var]] <- factor(combined_data[[cat_var]],
                                     levels = c("All",
unique(as.character(data_for_plot[[cat_var]]))))

  ggplot(combined_data, aes_string(x = cat_var, y = num_var, fill = cat_var)) +
    geom_boxplot(width = 0.6, outlier.size = 1) +
    scale_y_log10() +
    labs(x = '', y = sprintf('%s (log scale)', var_des), fill = 'Categories') +
    theme_minimal() +
    scale_fill_brewer(palette = 'Paired') +
    theme(legend.position = 'none',
          plot.title = element_text(size = 12, face = 'bold'),
          axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.text.x = element_blank(),
          y_limits <- c(0.001,0.005)) +
    ggtitle(sprintf('%s (%s)', var_des, num_var))}

p19 <- fx_plot.numoh(dw.v0, 'winewu', 'urbrur_all',
                     'Wine')
p20 <- fx_plot.numoh(dw.v0, 'nberwu', 'urbrur_all',
                     'Normal beer')
p21 <- fx_plot.numoh(dw.v0, 'spirwu', 'urbrur_all',
                     'Spirits') + labs(y = 'U / Week') +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'),
        axis.text.x = element_text(size = 12, face = 'bold',
                                   angle = 45, hjust = 1, vjust = 1))
p22 <- fx_plot.numoh(dw.v0, 'sberwu', 'urbrur_all',
                     'Strong beer') +
  theme(axis.text.x = element_text(size = 12, face = 'bold',
                                   angle = 45, hjust = 1, vjust = 1))
p23 <- fx_plot.numoh(dw.v0, 'sherwu', 'urbrur_all',
                     'Sherry') +
  theme(axis.text.x = element_text(size = 12, face = 'bold',
                                   angle = 45, hjust = 1, vjust = 1))
p24 <- fx_plot.numoh(dw.v0, 'popswu', 'urbrur_all',
                     'Alcopops') +
  theme(axis.text.x = element_text(size = 12, face = 'bold',
                                   angle = 45, hjust = 1, vjust = 1))
p25 <- fx_plot.numoh(dw.v0, 'drating', 'urbrur_all',
                   'Units alcohol/week') + labs(y = 'U / Week') +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
```

```
cat_labels <- c('no' = 'Below limit',
                'yes'= 'Above limit')
overlim15_plot <- fx_plot.cat(dw.v0, 'overlim15', 'urbrur_all',
                           'Weekly limits', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))

etoh_plots <- overlim15_plot + p25 + p19 + p20 + p21 + p22 + p23 + p24 +
  plot_layout(ncol = 4) +
  theme(legend.position = 'bottom',
        legend.text = element_text(size = 12, face = "bold"),
        legend.title = element_text(size = 12, face = "bold"))
print(etoh_plots)

#_______________________________________________________________________________
fx_plot_year <- function(data, var, varby, cat_labels = NULL) {

  data <- data %>%
    filter(!is.na(.data[[var]]), !is.na(.data[[varby]])) %>%
    count(.data[[var]], .data[[varby]]) %>%
    group_by(.data[[var]]) %>%
    mutate(total_count = sum(n),
           percent = n / total_count * 100) %>%
    ungroup()

  custom_labels <- function(x) {
    if (!is.null(cat_labels) && x %in% names(cat_labels)) {
      return(cat_labels[[x]])}
    x <- as.character(x)
    x <- gsub('_', ' ', x)
    x <- tools::toTitleCase(x)
    return(x)}

  ggplot(data, aes(x = .data[[var]], y = percent, group = .data[[varby]], color =
.data[[varby]])) +
    geom_line(size = 1) +
    geom_point(size = 3) +
    scale_y_continuous(limits = c(0, 100)) +
    labs(x = '', y = 'Percentage', color = 'Categories') +
    scale_color_brewer(palette = 'Paired') +
    scale_x_discrete(labels = function(x) sapply(x, custom_labels)) +
    theme_minimal() +
    theme(
      legend.position = 'bottom',
      plot.title = element_text(size = 14, face = 'bold'),
      axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
      axis.text.y = element_text(size = 12, face = 'bold'),
      axis.text.x = element_text(size = 12, angle = 45, hjust = 1, vjust = 1, face =
'bold')
    )
}

p0 <- fx_plot_year(dw.v0, 'syear', 'overlim15', cat_labels) +
  ggtitle('Survey Year (syear)')
print(p0)


#----------Visualisation by overlim15-------------------------------------------
# Define the plotting function with percentages

fx_plot.catover <- function(data, var, varby, cat_labels = NULL) {

  data <- data %>%
    filter(!is.na(.data[[var]]), !is.na(.data[[varby]])) %>%
    count(.data[[var]], .data[[varby]]) %>%
    group_by(.data[[var]]) %>%
    mutate(total_count = sum(n),
           percent = n / total_count * 100) %>%
    ungroup()

  custom_labels <- function(x) {
    if (!is.null(cat_labels) && x %in% names(cat_labels)) {
      return(cat_labels[[x]])}
    x <- as.character(x)
    x <- gsub('_', ' ', x)
    x <- toTitleCase(x)
    return(x)}

  ggplot(data, aes(x = .data[[var]],
                   y = percent, fill = .data[[varby]])) +
    geom_bar(stat = 'identity', position = 'dodge', color = 'black') +
    scale_y_continuous(limits = c(0, 100)) +
    labs(x = '', y = 'Percentage', fill = 'Categories') +
    scale_fill_brewer(palette = 'Paired') +
    scale_x_discrete(labels = function(x) sapply(x, custom_labels)) +
    theme_minimal() +
    theme(legend.position = 'none',
          plot.title = element_text(size = 12, face = 'bold'),
          axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
          axis.text.y = element_text(size = 12, face = 'bold'),
          axis.text.x = element_text(size = 12, angle = 45, hjust = 1, vjust = 1, face
= 'bold'))}

# Sociodemographic variables
cat_labels <- c('25_34' = '25 to 34',
                '35_44' = '35 to 44',
                '45_54' = '45 to 54',
                '55_64' = '55 to 64',
                'rest_UK' = 'Rest of the UK',
                'white_scottish' = 'White: Scottish',
                'white_restUK' = 'White: Other British',
                'white_other' = 'White: Other',
                'other_minority'= 'Other minority ethnic',
                'church_scotland' = 'Church of Scotland',
                'married_partner' = 'Married or civil partnership',
                'as_married' = 'Living as married',
                'divorced_dissolved' = 'Divorced or dissolved',
                'widowed' = 'Widowed or surviving partner')

# Plotting
p1 <- fx_plot.catover(dw.v0, 'ag16g10', 'overlim15', cat_labels) +
  ggtitle('Age (ag16g10)')

p2 <- fx_plot.catover(dw.v0, 'sex', 'overlim15')  +
  ggtitle('Sex (sex)')

p3 <- fx_plot.catover(dw.v0, 'birthpla3', 'overlim15', cat_labels) +
  ggtitle('Birthplace (birthpla3)')

p4 <- fx_plot.catover(dw.v0, 'ethnic05', 'overlim15',cat_labels) +
  ggtitle('Ethnic (ethnic05)')

p5 <- fx_plot.catover(dw.v0, 'religi04', 'overlim15', cat_labels) +
  ggtitle('Religion (religi04)')

p6 <- fx_plot.catover(dw.v0, 'maritalg', 'overlim15', cat_labels) +
  ggtitle('Marital (maritalg)')

dw.v0_sociover_plots <- p1 + p2 + p3 + p4 + p5 + p6 +
  plot_layout(ncol = 6) +
  plot_annotation(title = "A. OVERALL DATASET",
                  theme = theme(plot.title = element_text(size = 14, face = "bold",
hjust = 0)))
dw.v0_sociover_plots

#~~~~~~~~~~~~~~~~~~
# For urban
```

```r
u.p1 <- fx_plot.catover(dwu.v0, 'ag16g10', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
u.p2 <- fx_plot.catover(dwu.v0, 'sex', 'overlim15')
u.p3 <- fx_plot.catover(dwu.v0, 'birthpla3', 'overlim15', cat_labels)
u.p4 <- fx_plot.catover(dwu.v0, 'ethnic05', 'overlim15', cat_labels)
u.p5 <- fx_plot.catover(dwu.v0, 'religi04', 'overlim15',cat_labels)
u.p6 <- fx_plot.catover(dwu.v0, 'maritalg', 'overlim15', cat_labels)

dwu.v0_sociover_plots <- u.p1 + u.p2 + u.p3 + u.p4 + u.p5 + u.p6 +
  plot_layout(ncol = 6) +
  plot_annotation(title = "B. URBAN DATASET",
                  theme = theme(plot.title = element_text(size = 14, face = "bold",
hjust = 0)))
dwu.v0_sociover_plots

#~~~~~~~~~~~~~~~~~~~~~
#For rural
r.p1 <- fx_plot.catover(dwr.v0, 'ag16g10', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
r.p2 <- fx_plot.catover(dwr.v0, 'sex', 'overlim15')
r.p3 <- fx_plot.catover(dwr.v0, 'birthpla3', 'overlim15', cat_labels)
r.p4 <- fx_plot.catover(dwu.v0, 'ethnic05', 'overlim15', cat_labels)
r.p5 <- fx_plot.catover(dwr.v0, 'religi04', 'overlim15', cat_labels)
r.p6 <- fx_plot.catover(dwr.v0, 'maritalg', 'overlim15', cat_labels)

dwr.v0_sociover_plots <- r.p1 + r.p2 + r.p3 + r.p4 + r.p5 + r.p6 +
  plot_layout(ncol = 6) +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 12, face = "bold"),
        legend.title = element_text(size = 12, face = "bold"),
        plot.caption = element_text(size = 12, face = "bold")) +
  plot_annotation(title = "C. RURAL DATASET",
                  theme = theme(plot.title = element_text(size = 12, face = "bold",
hjust = 0))) +
  labs(caption = "Categories: no = Below the weekly limit, yes = Above the weekly
limit")
dwr.v0_sociover_plots

#~~~~~~~~~~~~~~~~~
# Health variables
print(unique(dw.v0$lifesat2))
cat_labels <- c('verygood_good' = 'Very good or good',
                'bad_verybad' = 'Bad or very bad')

p1 <- fx_plot.catover(dw.v0, 'genhelf', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold')) +
  ggtitle('General health  (genhelf)')

p2 <- fx_plot.catover(dw.v0, 'limitac_h', 'overlim15')  +
  ggtitle('LTC (limitac_h)')

p3 <- fx_plot.catover(dw.v0, 'lifesat2', 'overlim15',cat_labels) +
  ggtitle('Life satisfaction (lifesat2)')

p4 <- fx_plot.catover(dw.v0, 'adt10gptw', 'overlim15', cat_labels) +
  ggtitle('Activity level (adt10gptw)')

p5 <- fx_plot.catover(dw.v0, 'cig', 'overlim15', cat_labels) +
  ggtitle('Smoking (cig)')

dw.v0_healthover_plots <- p1 + p2 + p3 + p4 + p5 +
  plot_layout(ncol = 5) +
  plot_annotation(title = "(A) OVERALL DATASET",
                  theme = theme(plot.title = element_text(size = 14,
                                                          face = "bold", hjust = 0)))
dw.v0_healthover_plots
```

```
#~~~~~~~~~~~~~~~~~~
# For urban
u.p1 <- fx_plot.catover(dwu.v0, 'genhelf', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
u.p2 <- fx_plot.catover(dwu.v0, 'limitac_h', 'overlim15')
u.p3 <- fx_plot.catover(dwu.v0, 'lifesat2', 'overlim15', cat_labels)
u.p4 <- fx_plot.catover(dwu.v0, 'adt10gptw', 'overlim15',cat_labels)
u.p5 <- fx_plot.catover(dwu.v0, 'cig', 'overlim15', cat_labels)

dwu.v0_healthover_plots <- u.p1 + u.p2 + u.p3 + u.p4 + u.p5 +
  plot_layout(ncol = 5) +
  plot_annotation(title = "(B) URBAN DATASET",
                  theme = theme(plot.title = element_text(size = 14, face = "bold",
hjust = 0)))
dwu.v0_healthover_plots

#~~~~~~~~~~~~~~~~~~~~~
#For rural
r.p1 <- fx_plot.catover(dwr.v0, 'genhelf', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
r.p2 <- fx_plot.catover(dwr.v0, 'limitac_h', 'overlim15')
r.p3 <- fx_plot.catover(dwu.v0, 'lifesat2', 'overlim15', cat_labels)
r.p4 <- fx_plot.catover(dwr.v0, 'adt10gptw', 'overlim15', cat_labels)
r.p5 <- fx_plot.catover(dwr.v0, 'cig', 'overlim15', cat_labels)

dwr.v0_healthover_plots <- r.p1 + r.p2 + r.p3 + r.p4 + r.p5 +
  plot_layout(ncol = 5) +
  theme(legend.position = 'bottom') +
  plot_annotation(title = "(C) RURAL DATASET",
                  theme = theme(plot.title = element_text(size = 14,
                                                          face = "bold", hjust = 0)))
dwr.v0_healthover_plots

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Economic situation variables

p1 <- fx_plot.catover(dw.v0, 'hedqul08', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold')) +
  ggtitle('Educational qualification (hedqul08)')

p2 <- fx_plot.catover(dw.v0, 'schrpg7', 'overlim15')  +
  ggtitle('Classification HRP (schrpg7)')

p3 <- fx_plot.catover(dw.v0, 'neconacb', 'overlim15', cat_labels) +
  ggtitle('Economic activity (neconacb)')

p4 <- fx_plot.catover(dw.v0, 'simd20_rpa', 'overlim15', cat_labels) +
  ggtitle('SIMD 2020 (simd20_rpa)')

dw.v0_econover_plots <- p1 + p2 + p3 + p4 +
  plot_layout(ncol = 4) +
  plot_annotation(title = "(A) OVERALL DATASET",
                  theme = theme(plot.title = element_text(size = 14,
                                                          face = "bold", hjust = 0)))
dw.v0_econover_plots

#~~~~~~~~~~~~~~~~~~
# For urban
u.p1 <- fx_plot.catover(dwu.v0, 'hedqul08', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
u.p2 <- fx_plot.catover(dwu.v0, 'schrpg7', 'overlim15')
u.p3 <- fx_plot.catover(dwu.v0, 'neconacb', 'overlim15', cat_labels)
u.p4 <- fx_plot.catover(dwu.v0, 'simd20_rpa', 'overlim15', cat_labels)
```

```
dwu.v0_econover_plots <- u.p1 + u.p2 + u.p3 + u.p4 +
  plot_layout(ncol = 4) +
  plot_annotation(title = "(B) URBAN DATASET",
                  theme = theme(plot.title = element_text(size = 14, face = "bold",
hjust = 0)))
dwu.v0_econover_plots

#~~~~~~~~~~~~~~~~~~~~~
#For rural
r.p1 <- fx_plot.catover(dwr.v0, 'hedqul08', 'overlim15', cat_labels) +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold'))
r.p2 <- fx_plot.catover(dwr.v0, 'schrpg7', 'overlim15')
r.p3 <- fx_plot.catover(dwr.v0, 'neconacb', 'overlim15', cat_labels)
r.p4 <- fx_plot.catover(dwr.v0, 'simd20_rpa', 'overlim15', cat_labels)

dwr.v0_econover_plots <- r.p1 + r.p2 + r.p3 + r.p4 +
  plot_layout(ncol = 4) +
  theme(legend.position = 'bottom') +
  plot_annotation(title = "(C) RURAL DATASET",
                  theme = theme(plot.title = element_text(size = 14,
                                                          face = "bold", hjust = 0)))
dwr.v0_econover_plots

#~~~~~~~~~~~~~~~~~~~~~
# Alcohol consumtpion
fx_plot.numohover <- function(data, num_var, cat_var) {
   data_for_plot <- data %>%
    filter(!is.na(.data[[num_var]]), !is.na(.data[[cat_var]]))
  data_for_plot[[num_var]] <- ifelse(data_for_plot[[num_var]] == 0, 0.001,
data_for_plot[[num_var]])
  data_for_plot[[cat_var]] <- factor(data_for_plot[[cat_var]])
  ggplot(data_for_plot, aes_string(x = cat_var, y = num_var, fill = cat_var)) +
    geom_boxplot(width = 0.6, outlier.size = 1) +
    scale_y_log10() +
    labs(x = '', y = 'Units (log scale)', fill = 'Categories') +
    theme_minimal() +
    scale_fill_brewer(palette = 'Paired') +
    theme(legend.position = 'none',
          plot.title = element_text(size = 14, face = 'bold'),
          axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.text.x = element_blank(),
          axis.title = element_text(size = 10, face = 'bold'))}

# For overall
p1 <- fx_plot.numohover(dw.v0, 'winewu', 'overlim15') +
  ggtitle('Wine')

p2 <- fx_plot.numohover(dw.v0, 'nberwu', 'overlim15') +
  ggtitle('Normal beer')

p3 <- fx_plot.numohover(dw.v0, 'spirwu', 'overlim15') +
  ggtitle('Spirits')

p4 <- fx_plot.numohover(dw.v0, 'sberwu', 'overlim15')+
  ggtitle('Strong beer')

p5 <- fx_plot.numohover(dw.v0, 'sherwu', 'overlim15') +
  ggtitle('Sherry')

p6 <- fx_plot.numohover(dw.v0, 'popswu', 'overlim15')  +
  ggtitle('Alcopops')

p7 <- fx_plot.numohover(dw.v0, 'drating', 'overlim15') +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold')) +
  plot_annotation(title = "(A) OVERALL DATASET",
                  theme = theme(plot.title = element_text(size = 14,
```

```
                                                          face = "bold", hjust = 0))) +
  ggtitle('Units/week')

etoh_plots <- p7 + p1 + p2 + p3 + p4 + p5 + p6 +
  plot_layout(ncol = 7) +
  theme(legend.position = 'bottom')
print(etoh_plots)

# For urban
u.p1 <- fx_plot.numohover(dwu.v0, 'winewu', 'overlim15')
u.p2 <- fx_plot.numohover(dwu.v0, 'nberwu', 'overlim15')
u.p3 <- fx_plot.numohover(dwu.v0, 'spirwu', 'overlim15')
u.p4 <- fx_plot.numohover(dwu.v0, 'sberwu', 'overlim15')
u.p5 <- fx_plot.numohover(dwu.v0, 'sherwu', 'overlim15')
u.p6 <- fx_plot.numohover(dwu.v0, 'popswu', 'overlim15')
u.p7 <- fx_plot.numohover(dwu.v0, 'drating', 'overlim15') +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold')) +
  plot_annotation(title = "(B) URBAN DATASET",
                  theme = theme(plot.title = element_text(size = 14, face = "bold",
hjust = 0)))

u.etohover_plots <- u.p7 + u.p1 + u.p2 + u.p3 + u.p4 + u.p5 + u.p6 +
  plot_layout(ncol = 7)
print(u.etohover_plots)

# For rural
r.p1 <- fx_plot.numohover(dwr.v0, 'winewu', 'overlim15')
r.p2 <- fx_plot.numohover(dwr.v0, 'nberwu', 'overlim15')
r.p3 <- fx_plot.numohover(dwr.v0, 'spirwu', 'overlim15')
r.p4 <- fx_plot.numohover(dwr.v0, 'sberwu', 'overlim15')
r.p5 <- fx_plot.numohover(dwr.v0, 'sherwu', 'overlim15')
r.p6 <- fx_plot.numohover(dwr.v0, 'popswu', 'overlim15')
r.p7 <- fx_plot.numohover(dwr.v0, 'drating', 'overlim15') +
  theme(axis.title.y = element_text(size = 12, angle = 90, face = 'bold'),
        axis.text.y = element_text(size = 12, face = 'bold')) +
  plot_annotation(title = "(C) RURAL DATASET",
                  theme = theme(plot.title = element_text(size = 14, face = "bold",
hjust = 0)))

r.etohover_plots <- r.p7 + r.p1 + r.p2 + r.p3 + r.p4 + r.p5 + r.p6 +
  plot_layout(ncol = 7) +
  theme(legend.position = 'bottom')
print(r.etohover_plots)

#-------------------------------------------------------------------------------

# Dropping alcohol consumption units
names(dw.v0)

dw <- dw.v0[, -c(18:24)]
names(dw)
dim(dw)

#~~~~~~~~~~~~~~Missing data analysis~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Missing data stcs.
dw_na <- sapply(dw, function(x) sum(is.na(x)))
dw_nax100 <- sapply(dw, function(x) round(mean(is.na(x)) * 100, 2))
na_stats_df <- data.frame(Column = names(dw_na),
                          'Quantity' = dw_na,
                          'Percent' = dw_nax100)
na_stats_df.flex <- flextable(na_stats_df)
toword_fx(na_stats_df.flex, 'na_stats_df.docx')
na_stats_df.flex

# Complete cases in raw data
sum(is.na(dw))
dw_cc <- sum(complete.cases(dw))
dw_cc
```

```
dw_ccx100 <- dw_cc / nrow(dw)* 100
dw_ccx100

#Missing data: plots
miss_plt <- vis_miss(dw.raw)
miss_plt

miss_bar <- gg_miss_upset(dw.raw)
miss_bar

miss_plot <- gg_miss_var(dw) +
  theme(axis.text = element_text(size = 14),
        axis.title = element_text(size = 12))
miss_plot

#~~~~~~~~~~~~~~~~~~~~~~~~~~Recoding~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Re-coding ethnic05: ethnic group
levels(dw$ethnic05)
dw <- dw %>%
  mutate(ethnic05 = case_when(
    ethnic05 %in% c('white_other', 'other_minority', 'asian') ~ 'other',
    TRUE ~ as.character(ethnic05))) %>%
  mutate(ethnic05 = factor(
    ethnic05, levels = c('white_scottish', 'white_restuk', 'other')))
summary(dw$ethnic05)
table(dw$ethnic05, dw$urbrur_all)
tabyl(dw$ethnic05)
#~~~~~~~~~~~~~~~~~~~
# Re-coding religi04: religion
levels(dw$religi04)
dw <- dw %>%
  mutate(religi04 = case_when(
    religi04 %in% c('other_christian', 'another_religion') ~ 'other_religion',
    TRUE ~ as.character(religi04))) %>%
  mutate(religi04 = factor(
    religi04, levels = c('none', 'church_scotland', 'roman_catholic',
                         'other_religion')))
table(dw$religi04, dw$urbrur_all)
tabyl(dw$religi04)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Re-coding maritalg: marital status
levels(dw$maritalg)
dw <- dw %>%
  mutate(maritalg = case_when(
    maritalg %in% c('separated', 'divorced_dissolved', 'widowed') ~
'separated_widowed',
    TRUE ~ as.character(maritalg))) %>%
  mutate(maritalg = factor(
    maritalg, levels = c('married_partner', 'as_married', 'single',
                         'separated_widowed')))

table(dw$maritalg, dw$urbrur_all)
tabyl(dw$maritalg)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Re-coding genhelf: general health
levels(dw$genhelf)
dw <- dw %>%
  mutate(genhelf = case_when(
    genhelf %in% c('bad', 'very_bad') ~ 'bad_verybad',
    TRUE ~ as.character(genhelf))) %>%
  mutate(genhelf = factor(
    genhelf, levels = c('very_good', 'good', 'fair',
                         'bad_verybad')))

table(dw$genhelf, dw$urbrur_all)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# degree_or_higher: Undergraduate and Postgraduate degrees-SCQF Levels 9 to 12
```

```r
# HNC/D: Higher National Certificate/Diploma - SCQF Level 7 and 8
# Higher Grade (Higher) - SCQF Level 6

# Re-coding hedqul08: educational qualitication
levels(dw$hedqul08)
dw <- dw %>%
  mutate(hedqul08 = case_when(
    hedqul08 %in% c('standard_grade', 'other_school_level') ~ 'standard_school_grade',
    TRUE ~ as.character(hedqul08))) %>%
  mutate(hedqul08 = factor(
    hedqul08, levels = c('degree_or_higher', 'hnc_d', 'higher_grade',
                         'standard_school_grade', 'no_qualifications')))

table(dw$hedqul08, dw$urbrur_all)

# Re-coding schrpg7: social class
levels(dw.v0$schrpg7)
dw <- dw %>%
  mutate(schrpg7 = case_when(
    schrpg7 %in% c('unskilled_manual', 'others') ~ 'unskilled_other',
    TRUE ~ as.character(schrpg7))) %>%
  mutate(schrpg7 = factor(
    schrpg7, levels = c('professional', 'managerial_technical', 'skilled_non_manual',
                         'skilled_manual', 'semiskilled_manual', 'unskilled_other')))

table(dw$schrpg7, dw$urbrur_all)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Re-coding neconacb: economy activity
levels(dw$neconacb)

dw <- dw %>%
  mutate(neconacb = case_when(
    neconacb %in% c('ilo_unemployed','inactive') ~ 'unemployed_inactive',
    TRUE ~ as.character(neconacb))) %>%
  mutate(neconacb = factor(
    neconacb, levels = c('in_employment', 'unemployed_inactive')))

table(dw$neconacb, dw$urbrur_all)

# Re-coding adt10gptw: economy activity
levels(dw$adt10gptw)

dw <- dw %>%
  mutate(adt10gptw = case_when(
    adt10gptw %in% c('some_activity','low_activity') ~ 'low_activity',
    TRUE ~ as.character(adt10gptw))) %>%
  mutate(adt10gptw = factor(
    adt10gptw, levels = c('meets_recommendations', 'low_activity',
                          'very_low_activity')))

table(dw$adt10gptw, dw$urbrur_all)

summary(dw)
factor_fx(dw)

#------------------------------------------------------------------------------
sample_size_fx <- function(data, k, p) {
  factor_vars <- sapply(data, is.factor)
  levels_count <- sapply(data[, factor_vars], function(x) length(levels(x)))
  sample_size <- sum((levels_count - 1) * 10)
  peduzzi <- (10 * k) / p

  results <- tibble(
    Method = c("Sample_size", "Peduzzi"),
    Value = c(sample_size, peduzzi))

  return(results)}
```

```
result <- sample_size_fx(dw, 18, 0.2)
print(result)

#~~~~~~~~~~~~~~~~~Summary statistics after re-coding~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Descriptive statistics: after recording - rural vs urban (urbrur_all)
dw_stxurbrur <- dw %>%
  tbl_summary(by = urbrur_all,
              type = all_continuous() ~ 'continuous2',
              statistic = list(all_continuous() ~ c('{mean} (sd={sd})',
                                                    '{median} ({p25}, {p75})',
                                                    '{min} - {max}'),
                               all_categorical() ~ '{n} ({p}%)'),
              digits = all_continuous() ~ 2,
              missing = 'no') %>%
  add_overall() %>%
  add_n() %>%
  modify_header(label ~ '**Variable**') %>%
  modify_spanning_header(c('stat_1', 'stat_2') ~ '**Population Classification**') %>%
  bold_labels()
dw_stxurbrur.flex <- as_flex_table(dw_stxurbrur)
dw_stxurbrur.flex

toword_fx(dw_stxurbrur.flex, 'dw0_xurbrur.docx')

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Perform chi-square analysis with CrossTable
dw_cc <- dw[complete.cases(dw), ]
sum(is.na(dw_cc))
overlim_tab <- table(dw_cc$urbrur_all, dw_cc$overlim15)
overlim_tab
overlim_chisq <- chisq.test(overlim_tab)

expected_ok <- sum(overlim_chisq$expected >= 5) /
  length(overlim_chisq$expected) >= 0.8
overlimchisq_df <- data.frame(Statistic = round(overlim_chisq$statistic, 3),
                              DF = overlim_chisq$parameter,
                              p_value = round(overlim_chisq$p.value, 3),
                              Assumption_Met = ifelse(expected_ok, "Yes", "No"))
flextable(overlimchisq_df)

overlimchisqtab_df <- data.frame(
  Cells = apply(expand.grid(rownames(overlim_chisq$observed),
                            colnames(overlim_chisq$observed)),
                1, paste, collapse = " & "),
  Observed = as.vector(overlim_chisq$observed),
  Expected = as.vector(round(overlim_chisq$expected, 0)),
  Residuals = as.vector(round(overlim_chisq$residuals, 3)),
  Contribution =
as.vector(round((100*overlim_chisq$residuals^2/overlim_chisq$statistic), 2)))

overlimchisqtabflex <- flextable(overlimchisqtab_df)
overlimchisqtabflex
toword_fx(overlimchisqtabflex, 'overlimchisqtabflex.docx')

#Plot chi-squared result
corrplot(overlim_chisq$residuals, is.cor = FALSE,
         tl.cex = 1.2,
         col = COL1('Blues'), tl.col = 'black', cl.pos = 'b',
         addCoef.col = c('black', 'black', 'black', 'azure2'))

contrib <- 100*overlim_chisq$residuals^2/overlim_chisq$statistic
corrplot(contrib, is.cor = FALSE, method = 'color', tl.cex = 1.2,
         col = COL1('Blues'), tl.col = 'black', cl.pos = 'b',
         addCoef.col = c('black', 'black', 'black', 'azure2'))

#~~~~~~~~~~~~~~~~~~~~~Logisitic model - output variable: drkcat15~~~~~~~~~~~~~~
# Including non-drinkers dataset

#~~~~~~~~~~~~~~~~~~~~~~~Stepwise logistic regression~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```r
#Spliting the data into train and test subsets
dim(dw)
sum(is.na(dw))

set.seed(123)
training.samples <- dw$overlim15 %>%
  createDataPartition(p = 0.7, list = FALSE)

# Create the training and test sets
dw.trainv0  <- dw[training.samples, ]
dw.testv0 <- dw[-training.samples, ]
dim(dw.trainv0)
dim(dw.testv0)

# Complete cases in the training and test sets
dw.train <- dw.trainv0[complete.cases(dw.trainv0), ]
dw.test <- dw.testv0[complete.cases(dw.testv0), ]
dim(dw.train)
dim(dw.test)

dw$overlim15[dw.train$overlim15==0] <- "no"
dw$overlim15[dw.train$overlim15==1] <- "yes"
dw.train$overlim15 <- as.factor(dw.train$overlim15)
levels(dw.train$overlim15)

dw$overlim15[dw.test$overlim15==0] <- "no"
dw$overlim15[dw.test$overlim15==1] <- "yes"
dw.test$overlim15 <- as.factor(dw.test$overlim15)
levels(dw.test$overlim15)

tabyl(dw$overlim15)
tabyl(dw.train$overlim15)
tabyl(dw.test$overlim15)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
tabyl(dw$urbrur_all)
# Creating urban subset
dwu <- dw %>% filter(urbrur_all == 'urban')%>% dplyr::select(-urbrur_all)
dim(dwu)
# Urban: Training & Test
set.seed(123)
training.samples <- createDataPartition(dwu$overlim15, p = 0.7, list = FALSE)
dwu.trainv0  <- dwu[training.samples, ]
dwu.testv0 <- dwu[-training.samples, ]
dim(dwu.trainv0)
dim(dwu.testv0)

# Complete cases in the training and test sets
dwu.train <- dwu.trainv0[complete.cases(dwu.trainv0), ]
dwu.test <- dwu.testv0[complete.cases(dwu.testv0), ]
dim(dwu.train)
dim(dwu.test)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# # Creating rural subset
dwr <- dw %>% filter(urbrur_all == 'rural')%>% dplyr::select(-urbrur_all)
dim(dwr)
# Rural: Training & Test
set.seed(123)
training.samples <- createDataPartition(dwr$overlim15, p = 0.7, list = FALSE)
dwr.trainv0  <- dwr[training.samples, ]
dwr.testv0 <- dwr[-training.samples, ]
dim(dwr.trainv0)
dim(dwr.testv0)

# Complete cases in the training and test sets
dwr.train <- dwr.trainv0[complete.cases(dwr.trainv0), ]
dwr.test <- dwr.testv0[complete.cases(dwr.testv0), ]
```

```
dim(dwr.train)
dim(dwr.test)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
tabyl(dw.train$overlim15)
tabyl(dwu.train$overlim15)
tabyl(dwr.train$overlim15)

# Upsampling
upsampling_fx <- function(data, target_var, up) {
  up_sets <- list()
  set.seed(123)
  prop_ratios = c(0.3, 0.4, 0.5)

  for (prop_ratio in prop_ratios) {
    majority_data <- data[data[[target_var]] == 'no', ]
    minority_data <- data[data[[target_var]] == 'yes', ]

    target_minority_count <- ceiling(prop_ratio * nrow(majority_data)/(1-prop_ratio))
    upsampled_minority_data <- minority_data[sample(1:nrow(minority_data),
                                                    target_minority_count, replace =
TRUE), ]
    upsampled_data <- rbind(majority_data, upsampled_minority_data)
    dataset_name <- paste0('up', prop_ratio * 100, "_", gsub(".train", "",

deparse(substitute(data))))
    up_sets[[dataset_name]] <- upsampled_data}
  return(up_sets)}

# For overall population

updw_sets <- upsampling_fx(dw.train, 'overlim15')
sapply(updw_sets, function(df) {
  tbl <- table(df$overlim15)
  total <- sum(tbl)
  c(tbl, total = total)})
tabyl(updw_sets$up50_dw$overlim15)

# For urban population
updwu_sets <- upsampling_fx(dwu.train, 'overlim15')
sapply(updwu_sets, function(df) {
  tbl <- table(df$overlim15)
  total <- sum(tbl)
  c(tbl, total = total)})
tabyl(updwu_sets$up50_dw$overlim15)

# For rural population
updwr_sets <- upsampling_fx(dwr.train, 'overlim15')
sapply(updwr_sets, function(df) {
  tbl <- table(df$overlim15)
  total <- sum(tbl)
  c(tbl, total = total)})
tabyl(updwr_sets$up50_dwr$overlim15)
#_______________________________________________________________________________
# SMOTE Function
smote_fx <- function(data, output) {
  smote_sets <- list()
  set.seed(123)
  k_values <- c(3, 5, 10)
  over_ratios <- c(0.5, 0.75, 1)

  for (k in k_values) {
    for (over_ratio in over_ratios) {
      smote_data <- smotenc(data, var = output, k = k, over_ratio = over_ratio)
      dataset_name <- paste0('smk', k, 'r', over_ratio * 100, "_", gsub(".train", "",

deparse(substitute(data))))
      smote_sets[[dataset_name]] <- smote_data}}
```

```
  return(smote_sets)}

# For overall population
smdw_sets <- smote_fx(dw.train, "overlim15")
table(dwr$overlim15)
sapply(smdw_sets, function(df) {
  tbl <- table(df$overlim15)
  total <- sum(tbl)
  c(tbl, total = total)})


# For urban population
smdwu_sets <- smote_fx(dwu.train, "overlim15")
table(dwu$overlim15)
sapply(smdwu_sets, function(df) {
  tbl <- table(df$overlim15)
  total <- sum(tbl)
  c(tbl, total = total)})


# For rural population
smdwr_sets <- smote_fx(dwr.train, "overlim15")
table(dwr$overlim15)
sapply(smdwr_sets, function(df) {
  tbl <- table(df$overlim15)
  total <- sum(tbl)
  c(tbl, total = total)})

#_______________________________________________________________________________
# Plotting imbalance
imbalanceplot_fx <- function(data, title) {
  percent_imb <- data %>%
    group_by(overlim15) %>%
    summarise(count = n(), .groups = 'drop') %>%
    mutate(percentage = count / sum(count),
           source = "dw")

  ggplot(percent_imb, aes(x = overlim15, y = count, fill = overlim15)) +
    geom_bar(stat = "identity", position = "dodge") +
    geom_text(aes(label = scales::percent(percentage, accuracy = 0.1)),
              position = position_dodge(width = 0.5), vjust = -0.2,
              size = 6, color = "black") +
    scale_y_continuous(labels = scales::comma_format(), limits = c(0, 9000)) +
    labs(title = title,
         x = "overlim15",
         y = "Count") + theme_bw() +
    theme(legend.position = "",
          plot.title = element_text(size = 12, face = "bold"),
          axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks.y = element_blank(),
          axis.title.x = element_blank()) +
    scale_fill_brewer(palette = 'Paired')}

p1 <- imbalanceplot_fx(dw, 'A. ORIGINAL DATASET') +
  theme(axis.title.y = element_text(size = 12, face = 'bold', angle = 90),
        axis.text.y = element_text(),
        axis.ticks.y = element_line())

p2 <- imbalanceplot_fx(dw.train, 'B. TRAINING SUBSET')

plotxlist_fx <- function(datalist, prefix) {
  plots <- lapply(seq_along(datalist), function(i) {
    imbalanceplot_fx(datalist[[i]], paste0(prefix, i))})
  return(plots)}

updw_plots <- plotxlist_fx(updw_sets, 'C. sets UPSAMPLED')
updw_plots[[3]] <- updw_plots[[3]] +
  theme(axis.title.x = element_text(size = 12, face = 'bold'),
        axis.title.y = element_text(size =12, face = 'bold', angle = 90),
        axis.text.y = element_text(),
        axis.ticks.y = element_line())

smdw_plots <- plotxlist_fx(smdw_sets, 'C. sets SMOTE')
smdw_plots <- lapply(smdw_plots[1:3], function(plot) {
  plot + theme(axis.title.x = element_text(size = 12, face = 'bold'))})

imb_plots <- p1 + p2 + updw_plots[1:2] +
  updw_plots[3] +
  smdw_plots +
  plot_layout(ncol = 4)
imb_plots

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Function - stepwise logistic regression for each dataset
fx_steplg <- function(data) {
  full.model <- glm(overlim15 ~ ., data = data, family = binomial)
  step.model <- stepAIC(full.model, direction = 'both', trace = FALSE)
  formula_step <- formula(step.model)
  summary_step <- summary(step.model)
  list(formula = formula_step,
    summary = summary_step)}

# Stepwise - Original dataset
dw_sets <- setNames(list(dw.train, dwu.train, dwr.train),
                    c("dw.train", "dwu.train", "dwr.train"))


dw_step <- lapply(dw_sets, fx_steplg)
dw_step.form <- lapply(dw_step, function(res) res$formula)
print(dw_step.form)

# Stepwise - Upsampled dataset
updw_sets <- c(updw_sets, updwu_sets, updwr_sets)
names(updw_sets)

updw_step <- lapply(updw_sets, fx_steplg)
updw_step.form <- lapply(updw_step, function(res) res$formula)
print(updw_step.form)

# Stepwise - SMOTE dataset
smdw_sets <- c(smdw_sets, smdwu_sets, smdwr_sets)
names(smdw_sets)

smdw_step <- lapply(smdw_sets, fx_steplg)
smdw_step.form <- lapply(smdw_step, function(res) res$formula)
print(smdw_step.form)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Cross validation - Function
fiveStats <- function(...) c(twoClassSummary(...),
                             defaultSummary(...))

cv_fx <- function(formula, data) {
  set.seed(123)
  control <- trainControl(method = "repeatedcv", number = 10, repeats = 5,
                          savePredictions = TRUE,
                          classProbs = TRUE,
                          summaryFunction = fiveStats)

  model_cv <- train(formula, data = data, method = "glm",
                 family = binomial,
                 trControl = control,
                 metric = "ROC")
  return(model_cv)}

#________________________________________________________________________________
```

```r
# Cross validation Metrics - Function
metrics_fx <- function(cv_res, datasets) {
  metrics_list <- lapply(seq_along(cv_res), function(i) {

    model_cv <- cv_res[[i]]
    res <- model_cv$results
    res$Dataset <- names(datasets)[i]
    return(res)})

  combined_metrics <- do.call(rbind, metrics_list)
  flex_table <- flextable(combined_metrics[,-c(1, 6:11)])
  return(flex_table)}

#_______________________________________________________________________________
# Cross validation Coefficients and p-values Function
coef_fx <- function(cv_res, datasets) {
  coef_list <- lapply(seq_along(cv_res), function(i) {

  model <- cv_res[[i]]$finalModel
  tidy_model <- tidy(model)
  tidy_model$Dataset <- names(datasets)[i]
  return(tidy_model)})

  dw_cv.resall <- do.call(rbind, coef_list)
  dw_cv.resall$p.value <- round(dw_cv.resall$p.value, 2)
  dw_cv.resallflex <- flextable(dw_cv.resall) %>%
    theme_vanilla() %>% autofit() %>%
    color(i = ~ p.value < 0.05, j = "p.value", color = "darkgreen")%>%
    color(i = ~ p.value > 0.05, j = "p.value", color = "red3") %>%
    color(i = ~ p.value == 0.05, j = "p.value", color = "darkorange") %>%
    bold(i = ~ p.value < 0.05 | p.value == 0.05, j = "p.value", bold = TRUE)
  return(dw_cv.resallflex)}

#_______________________________________________________________________________
# For original sets
dw_cv <- mapply(cv_fx, dw_step.form, dw_sets, SIMPLIFY = FALSE)

# Metrics & Coef.: dw_cv
dw_cv.metricsflex <- metrics_fx(dw_cv, dw_sets)
dw_cv.metricsflex
toword_fx(dw_cv.metricsflex, 'cv.metrics.docx')

dw_cv.resallflex <- coef_fx(dw_cv, dw_sets)
dw_cv.resallflex
toword_fx(dw_cv.resallflex, 'dw_cv.resall.docx')

#_______________________________________________________________________________
# For upsampled sets
updw_cv <- mapply(cv_fx, updw_step.form, updw_sets, SIMPLIFY = FALSE)

# Metrics & Coef.: updw_cv
updw_cv.metricsflex <- metrics_fx(updw_cv, updw_sets)
updw_cv.metricsflex
toword_fx(updw_cv.metricsflex, 'updw_cv.metrics.docx')

updw_cv.resallflex <- coef_fx(updw_cv, updw_sets)
updw_cv.resallflex
toword_fx(updw_cv.resallflex, 'updw_cv.resall.docx')

#_______________________________________________________________________________
# For SMOTE sets
smdw_cv <- mapply(cv_fx, smdw_step.form, smdw_sets, SIMPLIFY = FALSE)

# Metrics & Coef.: smdw_cv
smdw_cv.metricsflex <- metrics_fx(smdw_cv, smdw_sets)
smdw_cv.metricsflex
toword_fx(smdw_cv.metricsflex, 'smdw_cv.metrics.docx')

smdw_cv.resallflex <- coef_fx(smdw_cv, smdw_sets)
smdw_cv.resallflex
toword_fx(smdw_cv.resallflex, 'smdw_cv.resall.docx')

#_______________________________________________________________________________
# Fitting the model & Evaluating the model
#_______________________________________________________________________________
# Function: evaluating the model
mod_glm.ev <- function(model, data, threshold) {
  pre_prob <- predict(model, data, type = "response")
  pre_class <- ifelse(pre_prob > threshold, "yes", "no")
  tab_mod <- table(predicted = pre_class, observed = data$overlim15)
  sens_mod <- (tab_mod[2, 2] / sum(tab_mod[, 2])) * 100
  spec_mod <- (tab_mod[1, 1] / sum(tab_mod[, 1])) * 100
  correct_mod <- sum(diag(tab_mod)) / sum(tab_mod) * 100
  precision_mod <- (tab_mod[2, 2] / sum(tab_mod[2, ])) * 100
  f1_mod <- 2 * (precision_mod * sens_mod) / (precision_mod + sens_mod)
  flex_table <- flextable(data.frame(Metric = c('Sensitivity', 'Specificity',
                                                'Accuracy', 'Precision',
                                                'F1 Score'),
                                     Value = round(c(sens_mod, spec_mod,
                                                     correct_mod, precision_mod,
                                                     f1_mod), 2)))
  return(flex_table)}


#_______________________________________________________________________________
# Models to evaluate in test subsets - Urban & Rural
#_______________________________________________________________________________
#  Urban & Rural
# Model: SMOTE k=10 / r = 100
dw.sm_glm <- glm(smdw_step.form$smk10r100_dw,
                  data = smdw_sets$smk10r100_dw,
                  family = binomial)

summary(dw.sm_glm)
print(dw.sm_glm)
variables <- labels(terms(dw.sm_glm))
print(variables)

dw.sm_glm.roc <- ROC(predict(dw.sm_glm, newdata = dw.test,
                              type = "response"),
                      dw.test$overlim15,
                      plot = "ROC")

threshold = 0.437
dw.sm_glm.metrics <- mod_glm.ev(dw.sm_glm, dw.test, threshold)
dw.sm_glm.metrics
toword_fx(dw.sm_glm.metrics, 'dw.sm_glm.metrics.docx')

#__________________________________________________
# Urban population
# Model: SMOTE k=10 / r = 100
dwu.sm_glm <- glm(smdw_step.form$smk10r100_dwu,
                  data = smdw_sets$smk10r100_dwu,
                  family = binomial)

summary(dwu.sm_glm)
print(dwu.sm_glm)
variables <- labels(terms(dwu.sm_glm))
print(variables)

dwu.sm_glm.roc <- ROC(predict(dwu.sm_glm, newdata = dwu.test,
                              type = "response"),
                      dwu.test$overlim15,
                      plot = "ROC")

threshold = 0.416
dwu.sm_glm.metrics <- mod_glm.ev(dwu.sm_glm, dwu.test, threshold)
dwu.sm_glm.metrics
```

```
toword_fx(dwu.sm_glm.metrics, 'dwu.sm_glm.metrics.docx')

#________________________________________________
# # For RURAL population - SMOTE data k=5 or=1
# Model: dwr
dwr.sm_glm <- glm(smdw_step.form$smk5r100_dwr,
                  data = smdw_sets$smk5r100_dwr,
                  family = binomial)


summary(dwr.sm_glm)
print(dwr.sm_glm)
variables <- labels(terms(dwr.sm_glm))
print(variables)

dwr.sm_glm.roc <- ROC(predict(dwr.sm_glm, newdata = dwr.test,
                              type = "response"),
                      dwr.test$overlim15,
                      plot = "ROC")

threshold = 0.371
dwr.sm_glm.metrics <- mod_glm.ev(dwr.sm_glm, dwr.test, threshold)
dwr.sm_glm.metrics


toword_fx(dwr.sm_glm.metrics, 'dwr.sm_glm.metrics.docx')

#_______________________________________________________________________________
# Interpreting the model
#_______________________________________________________________________________

# Function: output logistic models
outmod_fx <- function(mod) {
  summary_mod <- summary(mod)
  coefficients <- summary_mod$coefficients[, 1:4]
  odds_ratio <- exp(coefficients[, 'Estimate'])
  lower_ci <- exp(coefficients[, 'Estimate'] - 1.96 * coefficients[, 'Std. Error'])
  upper_ci <- exp(coefficients[, 'Estimate'] + 1.96 * coefficients[, 'Std. Error'])

  summary_df <- data.frame(
    Variable = rownames(coefficients),
    Estimate = round(coefficients[, 'Estimate'],3),
    'Std. Error' = round(coefficients[, 'Std. Error'],3),
    'Z-value' = round(coefficients[, 'z value'],3),
    'P-value' = format(round(coefficients[, 'Pr(>|z|)'],3)),
    'Odds Ratio' = round(odds_ratio, 3),
    'CI Lower' = round(lower_ci, 3),
    'CI Upper' = round(upper_ci, 3))

  ft <- flextable(summary_df)
  ft <- color(ft, j = 'P.value', i = ~ P.value < 0.05, color = 'red')
  return(list(ft = ft, summary_df = summary_df))}

# # Final model: Overall - ODDs / CI
dw_glmcoef <- outmod_fx(dw.sm_glm)
dw_glmcoef$summary_df
toword_fx(dw_glmcoef$ft, 'dw_glmcoef.docx')


# Final model: Urban - ODDs / CI
dwu_glmcoef <- outmod_fx(dwu.sm_glm)
dwu_glmcoef$summary_df
toword_fx(dwu_glmcoef$ft, 'dwu_glmcoef.docx')

# Final model: Rural - ODDs / CI
dwr_glmcoef <- outmod_fx(dwr.sm_glm)
dwr_glmcoef$summary_df
```

```
toword_fx(dwr_glmcoef$ft, 'dwr_glmcoef.docx')

# OR and CI plots
plot_or_ci <- function(df, title, colour) {
  ggplot(df, aes(x = reorder(Variable, Odds.Ratio), y = Odds.Ratio)) +
    geom_point(color = colour, size = 2) +
    geom_errorbar(aes(ymin = CI.Lower, ymax = CI.Upper), width = 0.2, color =
"slategray") +
    coord_flip() +
    theme_minimal() +
    labs(title = title, y = "Odds Ratio (log scale)", x = "Variable") +
    theme(axis.text.y = element_text(size = 10, face = "bold", color = "black"),
      axis.text.x = element_text(size = 10, face = "bold"),
      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
      axis.title.x = element_text(size = 12, face = "bold"),
      axis.title.y = element_text(size = 12, face = "bold")) +
    geom_hline(yintercept = 1, linetype = "dashed", color = "red")}



dw_odd <- plot_or_ci(dw_glmcoef$summary_df, "Urban & Rural Model", "steelblue3")
dwu_odd <- plot_or_ci(dwu_glmcoef$summary_df, "Urban Model", "navy")
dwr_odd <- plot_or_ci(dwr_glmcoef$summary_df, "Rural Model", "yellowgreen")

print(dw_odd)
print(dwu_odd)
print(dwr_odd)

print(smdw_step.form$smk5r100_dwr)
print(smdw_step.form$smk10r100_dwu)

#_______________________________________________________________________________
#Random Forest
#_______________________________________________________________________________
# Target and features
install.packages('ranger')
library(ranger)
# Rural datasets
dwr_datasets <- list(dwr.train = dwr.train,
                     up30_dwr = updwr_sets$up30_dwr,
                     up40_dwr = updwr_sets$up40_dwr,
                     up50_dwr = updwr_sets$up50_dwr,
                     smk3r50_dwr = smdwr_sets$smk3r50_dwr,
                     smk3r75_dwr = smdwr_sets$smk3r75_dwr,
                     smk3r100_dwr = smdwr_sets$smk3r100_dwr,
                     smk5r50_dwr = smdwr_sets$smk5r50_dwr,
                     smk5r75_dwr = smdwr_sets$smk5r75_dwr,
                     smk5r100_dwr = smdwr_sets$smk5r100_dwr,
                     smk10r50_dwr = smdwr_sets$smk10r50_dwr,
                     smk10r75_dwr = smdwr_sets$smk10r75_dwr,
                     smk10r100_dwr = smdwr_sets$smk10r100_dwr)

# Function to split output from features
splitrf_fx <- function(data_list) {
  result <- list()

  for (name in names(data_list)) {
    data <- data_list[[name]]

    X_train <- data %>% dplyr::select(-overlim15)
    y_train <- data$overlim15

    result[[name]] <- list(
      X_train = X_train,
      y_train = y_train)}

  return(result)}

# Splitting data - RF
```

```
dwr_splitrf <- splitrf_fx(dwr_datasets)

# Best hyperparameters
tune_random_forest <- function(X_train, y_train) {
  seed = 123
  hyper_grid <- expand.grid(
    n_trees       = c(100, 200, 300, 500),
    mtry          = seq(2, 16, by = 1),
    node_size     = seq(3, 9, by = 2),
    sampe_size    = c(.5, .6, .7, .8, 1),
    OOB_RMSE      = 0)

  for (i in 1:nrow(hyper_grid)) {
    model <- ranger(
      seed            = 123,
      formula         = y_train ~ .,
      data            = cbind(X_train, y_train),
      num.trees       = hyper_grid$n_trees[i],
      mtry            = hyper_grid$mtry[i],
      min.node.size   = hyper_grid$node_size[i],
      sample.fraction = hyper_grid$sampe_size[i])

    hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)}

  best_hyperparameters <- hyper_grid[which.min(hyper_grid$OOB_RMSE), ]
  return(best_hyperparameters)}

best_params_list <- lapply(dwr_splitrf, function(dataset) {
  tune_random_forest(dataset$X_train, dataset$y_train)})
best_params_list

# Cross validation
tune_grid <- expand.grid(
  .mtry = 6,
  .splitrule = "gini",
  .min.node.size = 3)

control <- trainControl(method = "repeatedcv", number = 10,
                        repeats = 5,
                        savePredictions = TRUE,
                        classProbs = TRUE,
                        summaryFunction = fiveStats)

set.seed(123)
cv_model <- train(
  x = dwr_splitrf$up50_dwr$X_train,
  y = dwr_splitrf$up50_dwr$y_train,
  method = "ranger",
  trControl = control,
  tuneGrid = tune_grid,
  num.trees = 500,
  sample.fraction = 1)

print(cv_model)
rf_results <- as.data.frame(cv_model$results)
rf_results <- flextable(rf_results)
toword_fx(rf_results, 'rf_results.docx')

#Final model
rf_model <- ranger(
  formula = y ~  .,
  data = cbind(dwr_splitrf$up50_dwr$X_train, y = dwr_splitrf$up50_dwr$y_train),
  num.trees = 500,
  mtry = 6,
  min.node.size = 3,
  sample.fraction = 1,
  importance = 'permutation',
  local.importance = TRUE,
  scale.permutation.importance = TRUE)
```

```
rf_model$variable.importance
print(rf_model)
summary(rf_model)
rf_model$variable.importance
var_rf <- vip(rf_model)
var_rf

# Evaluating in test subset
predic_rf <- predict(rf_model, data = dwr.test)$predictions

tab_rf <- table(predicted = predic_rf, observed = dwr.test$overlim15)

sens_rf <- (tab_rf[2, 2] / sum(tab_rf[, 2])) * 100
spec_rf <- (tab_rf[1, 1] / sum(tab_rf[, 1])) * 100
correct_rf <- (sum(diag(tab_rf)) / sum(tab_rf)) * 100
precision_rf <- (tab_rf[2, 2] / sum(tab_rf[2, ])) * 100
f1_rf <- 2 * (precision_rf * sens_rf) / (precision_rf + sens_rf)

metricsflex_rf <- flextable(data.frame(
  Metric = c('Sensitivity', 'Specificity', 'Accuracy', 'Precision', 'F1 Score'),
  Value = round(c(sens_rf, spec_rf, correct_rf, precision_rf, f1_rf), 2)))

metricsflex_rf
toword_fx(metricsflex_rf, 'metricsflex_rf.docx')

library(vip)

print(dw_step.form)
print(dw)
```