```r
#' Weight a survey using post-stratification with weighting indicator(s) from
#' ACS tables
#'
#' @param mysurvey A survey data frame, such as that created by
#' \code{simulate_survey}
#' @param ... Weighting indicator(s) to be used for post-stratification.
#' One or more of \code{sex}, \code{raceethnicity}, \code{age}, and/or
#' \code{education} which must be columns in the survey data frame. Both
#' \code{age} and \code{education} cannot be used for post-stratification at the
#' same time because of how the ACS tables are organized.
#' @param dots List of weighting indicator(s) as string(s)
#' @param force_edu Should the weighting be done with the ACS education tables?
#' Defaults to \code{FALSE}; cannot be \code{TRUE} if \code{age} is one of the
#' weighting indicators
#'
#' @return The original survey data frame with 1 column added, \code{weight},
#' the post-stratification weight for each row in the survey.
#'
#' @details \code{weight_wwc} is given bare names while \code{weight_wwc_} is
#' given strings and is therefore suitable for programming with. One column of
#' \code{mysurvey} must be \code{geography}, to indicate what ACS data to use
#' for post-stratification weighting.
#'
#' @import dplyr
#' @importFrom reshape2 melt
#' @importFrom stats as.formula
#' @importFrom stats complete.cases
#'
#' @name weight_wwc
#'
#' @examples
#' data(texassurvey)
#' weight_wwc(texassurvey, sex, raceethnicity)
#' data(twostatessurvey)
#' weight_wwc(twostatessurvey, sex, education)
#'
#' @export
weight_wwc <- function(mysurvey, ..., force_edu = FALSE) {
        # NSE magic
        dots <- eval(substitute(alist(...)))
        dots <- purrr::map(dots, col_name)

        weight_wwc_(mysurvey, dots, force_edu)
}

#' @rdname weight_wwc
#' @export
weight_wwc_ <- function(mysurvey, dots, force_edu = FALSE) {

        # error handling for weighting indicator
        if (any(purrr::map(dots, function(x)
        {x[[1]] %in% c("sex", "raceethnicity", "age", "education")}) == FALSE)) {
                stop("indicators must be one or more of sex, raceethnicity, age, and education") }
        if (sum(purrr::map_lgl(dots, function(x) {x[[1]] %in% c("age", "education")})) > 1) {
                stop("indicators cannot include both age and education") }
        if ("age" %in% dots & force_edu) {
                stop("force_edu cannot be TRUE if age is one of the indicators") }

        # exclude rows/observations/respondents who have NA for geography
        mysurvey <- mysurvey[!is.na(mysurvey$geography),]

        # what is the population of the survey by geography?
        surveytotals <- mysurvey %>% group_by(geography) %>%
                summarise(surveytotal = n())
        totalsurvey <- nrow(mysurvey)

        # download and process ACS data
        geovector <- mysurvey %>% distinct(geography)
        if ("education" %in% dots | force_edu) {
                acsDF <- acsedutable %>%
                        filter(region %in% geovector$geography)
        } else {
                acsDF <- acsagetable %>%
                        filter(region %in% geovector$geography)
        }
        totalpop <- as.numeric(acsDF %>% distinct(geototal) %>%
                                       summarise(sum = sum(geototal)))


        # find the relative weights for each geography in the survey
        totals <- acsDF %>% group_by(region) %>% distinct(geototal) %>%
                ungroup %>%
                left_join(surveytotals, by = c("region" = "geography")) %>%
                mutate(geototal = geototal/totalpop,
                       surveytotal = surveytotal/totalsurvey,
                       geoweight = geototal/surveytotal)

        # separate the survey into geographical groups, then find weights
        mysurvey <- mysurvey %>% group_by(geography) %>% tidyr::nest()
        acsDF <- acsDF %>% group_by(region) %>% tidyr::nest()
        nestedDF <- left_join(mysurvey, acsDF,
                              by = c("geography" = "region")) %>%
                rename(mysurvey = data.x, acsDF = data.y)

        ret <- purrr::map2_df(nestedDF$mysurvey, nestedDF$acsDF, weight_skeleton_, dots,
                              .id = "geography") %>%
                mutate(geography = as.character(factor(geography,
                                                        labels = nestedDF$geography))) %>%
                left_join(totals, by = c("geography" = "region")) %>%
                mutate(weight = weight * geoweight) %>% # weight each respondent by geography
                select(-geototal, -surveytotal, -geoweight)
        ret
}
```