# Tidy EDA and Statistical Concepts

Claus C Pörtner

2018-10-08

# Today

# Agenda

1. Getting data ready
2. Exploratory data analysis
3. Simple statistics and tests

# Analysis ALWAYS begins with EDA

Goal of EDA is to

- Develop understanding of data
- Detection of mistakes
- Assessing direction and rough size of relationships
- Preliminary selection of appropriate models

# Tidyverse

# Tidyverse???

A coherent set of packages in R
- Share a common data representation and API design

Supported and promoted by RStudio
- Help materials on RStudio

Supports the end-to-end workflow for a variety of analysis goals

# But first, packages

Use either

- `install.packages("tidyverse")`
- Click Tools -> Install Packages…
- Packages tab in RStudio

```
# install.packages("tidyverse")
```

## Also need to load the library:

```
# The tidyverse package contains ggplot2, tibble, tidyr,
# readr, purr, and dplyr and several others
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
```

```
## ✔ ggplot2 3.0.0     ✔ purrr   0.2.5
## ✔ tibble  1.4.2     ✔ dplyr   0.7.6
## ✔ tidyr   0.8.1     ✔ stringr 1.3.1
## ✔ readr   1.1.1     ✔ forcats 0.3.0
```

```
## Warning: package 'dplyr' was built under R version 3.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

# An aside on file paths

I use the package `here` to ease navigation

```
library(here)
```

```
## here() starts at /Users/claus/teaching/econ5100/2018-09/git_5100_2018_09
```

If you have a R project set up then you can, for example, use

```
here("raw_data", "my_data.csv")
```

to refer to your data, no matter where you are!

# Other packages

```
# The gridExtra package contains grid.arrange function used
# to combine plots
library(gridExtra)

# The GGally package contains ggpairs which is a custom
# correlation graph for ggplot2
library(GGally)
```

mul

# A Tidy Data Example

# Definition?

1. Each variable forms a column
2. Each observation forms a row
3. Each value is a cell

# Not far off!

Classroom data are like teddy bears; real data are like a grizzly with salmon blood dripping out its mouth

— Jenny Bryan

# Today's Messy Data

2005-2006 National Family and Health Survey from India

Download `nfhs_3.csv` from Canvas

```
nfhs <- read_csv(here("raw_data", "nfhs_3.csv"))
```

# What is in the data?

Problem: most real data sets are large

(This is a subset of ~3,500 variables!)

First step: check variables

Standard approach:

- Click on chevron (environment)
- `View(nfhs)`
- `head(nfhs)`
- `str(nfhs)`

# What happens?

```
str(nfhs)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    109041 obs. of  234 variables:
##  $ hhid    : chr  "28  1   1" "28  1   2" "28  1   3" "28  1   4" ...
##  $ hv000   : chr  "IA5" "IA5" "IA5" "IA5" ...
##  $ hv001   : int  28001 28001 28001 28001 28001 28001 28001 28001 28001 28001 ...
##  $ hv002   : int  1 2 3 4 6 7 8 10 11 13 ...
##  $ hv003   : int  2 1 1 1 1 2 6 3 2 1 ...
##  $ hv004   : chr  NA NA NA NA ...
##  $ hv005   : int  2506085 2506085 2506085 2506085 2506085 2506085 2506085 2506085 2506085 250608
##  $ hv006   : int  1 3 1 1 1 1 1 1 1 3 ...
##  $ hv007   : int  2006 2006 2006 2006 2006 2006 2006 2006 2006 2006 ...
##  $ hv008   : int  1273 1275 1273 1273 1273 1273 1273 1273 1273 1275 ...
##  $ hv009   : int  6 5 1 5 7 3 8 6 2 2 ...
##  $ hv010   : int  2 2 0 1 1 1 3 1 1 1 ...
##  $ hv011   : int  1 1 0 0 2 1 2 2 1 0 ...
##  $ hv012   : int  6 5 1 5 7 3 8 6 2 2 ...
##  $ hv013   : int  6 5 1 4 7 3 8 6 2 2 ...
##  $ hv014   : int  2 0 0 0 2 0 2 1 0 0 ...
##  $ hv015   : chr  "completed" "completed" "completed" "completed" ...
##  $ hv016   : int  28 26 28 27 27 26 26 26 28 26 ...
##  $ hv017   : int  1 4 1 1 1 1 1 1 2 2 ...
##  $ hv018   : int  142 104 104 131 131 142 104 131 103 104 ...
##  $ hv019   : int  24 12 24 24 24 24 24 24 24 12 ...
##  $ hv020   : chr  "all woman sample" "all woman sample" "all woman sample" "all woman sample" .
```

# Alternative: get column names

```
colnames(nfhs) # or names()
```

```
##   [1] "hhid"     "hv000"    "hv001"    "hv002"    "hv003"    "hv004"
##   [7] "hv005"    "hv006"    "hv007"    "hv008"    "hv009"    "hv010"
##  [13] "hv011"    "hv012"    "hv013"    "hv014"    "hv015"    "hv016"
##  [19] "hv017"    "hv018"    "hv019"    "hv020"    "hv021"    "hv022"
##  [25] "hv023"    "hv024"    "hv025"    "hv026"    "hv027"    "hv028"
##  [31] "hv030"    "hv031"    "hv032"    "hv033"    "hv035"    "hv040"
##  [37] "hv041"    "hv042"    "hv043"    "hv044"    "hv201"    "hv202"
##  [43] "hv204"    "hv205"    "hv206"    "hv207"    "hv208"    "hv270"
##  [49] "hvidx_01" "hvidx_02" "hvidx_03" "hvidx_04" "hvidx_05" "hvidx_06"
##  [55] "hvidx_07" "hvidx_08" "hvidx_09" "hvidx_10" "hvidx_11" "hvidx_12"
##  [61] "hvidx_13" "hvidx_14" "hvidx_15" "hvidx_16" "hvidx_17" "hvidx_18"
##  [67] "hvidx_19" "hvidx_20" "hvidx_21" "hvidx_22" "hvidx_23" "hvidx_24"
##  [73] "hvidx_25" "hvidx_26" "hvidx_27" "hvidx_28" "hvidx_29" "hvidx_30"
##  [79] "hvidx_31" "hvidx_32" "hvidx_33" "hvidx_34" "hvidx_35" "hv108_01"
##  [85] "hv108_02" "hv108_03" "hv108_04" "hv108_05" "hv108_06" "hv108_07"
##  [91] "hv108_08" "hv108_09" "hv108_10" "hv108_11" "hv108_12" "hv108_13"
##  [97] "hv108_14" "hv108_15" "hv108_16" "hv108_17" "hv108_18" "hv108_19"
## [103] "hv108_20" "hv108_21" "hv108_22" "hv108_23" "hv108_24" "hv108_25"
## [109] "hv108_26" "hv108_27" "hv108_28" "hv108_29" "hv108_30" "hv108_31"
## [115] "hv108_32" "hv108_33" "hv108_34" "hv108_35" "ha0_01"   "ha0_02"
## [121] "ha0_03"   "ha0_04"   "ha0_05"   "ha0_06"   "ha0_07"   "ha0_08"
## [127] "ha0_09"   "ha0_10"   "ha0_11"   "ha1_01"   "ha1_02"   "ha1_03"
## [133] "ha1_04"   "ha1_05"   "ha1_06"   "ha1_07"   "ha1_08"   "ha1_09"
```

# Variables

hhid-hv208: Household level variables

hvidx_XX Household roster

hv108_XX: Education in years

haX_XX Female roster with age, height, and weight

hbX_XX Male roster with age, height, and weight

# Is this tidy?

Depends!

When?

Here we want individual level obs with household level information

Strategy?

# Household level information

```
# Range of variables (be careful!)
hh <- select(nfhs, hhid:hv208, hv270)
```

# Education

```
educ <- select(nfhs, hhid,
               starts_with("hvidx"),
               contains("hv108"))
```

# Female information

```
# Note RegEx needs to be escaped
female <- select(nfhs, hhid,
                 matches("ha\\d_\\d\\d")
                 )
```

# Male information

```
male <- select(nfhs, hhid,
               contains("hb")
               )
```

# We need some tidyr

`gather()` takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer.

`spread()` takes two columns (key & value) and spreads in to multiple columns, it makes “long” data wider.

`separate()` turns a single character column into multiple columns

`united()` paste together multiple columns into one

# Gathering (wide -> long)

This is very painful since you cannot do it in one step

One option is this:

```
educ_long1 <- gather(educ, variable_name,
                     var_value, -hhid)

educ_long2 <- separate(educ_long1, variable_name,
                       c("var", "number"), sep = "_")

educ_long3 <- spread(educ_long2, key = var, value = var_value)

educ_long4 <- filter(educ_long3, !is.na(hvidx))
```

Why is this not a good approach (programming wise)?

Better, but not exactly easy to ready:

```
educ <- gather(educ, variable_name, var_value, -hhid)

educ <- separate(educ, variable_name, c("var", "number"), sep = "_")

educ <- spread(educ, key = var, value = var_value)

educ <- filter(educ, !is.na(hvidx))
```

# %>% operator…

learn it, love it, leverage it

```
educ <- nfhs %>%
  select(hhid, starts_with("hvidx"), contains("hv108")) %>%
  gather(variable_name, var_value, -hhid) %>%
  separate(variable_name, c("var", "number"), sep = "_") %>%
  spread(key = var, value = var_value) %>%
  filter(!is.na(hvidx)) %>%
  select(-number) %>%
  rename(roster_id = hvidx, educ = hv108) # Something to merge on!
```

# Need to do the same for female…

```
female <- nfhs %>%
  select(hhid, matches("ha\\d_\\d\\d")) %>%
  gather(variable_name, var_value, -hhid) %>%
  separate(variable_name, c("var", "number"), sep = "_") %>%
  spread(key = var, value = var_value) %>%
  filter(!is.na(ha0)) %>%
  select(-number) %>%
  rename(roster_id = ha0, age = ha1, weight = ha2, height = ha3) %>%
  mutate(female = TRUE)
```

# …and male

```
male <- nfhs %>%
  select(hhid, contains("hb")) %>%
  gather(variable_name, var_value, -hhid) %>%
  separate(variable_name, c("var", "number"), sep = "_") %>%
  spread(key = var, value = var_value) %>%
  filter(!is.na(hb0)) %>%
  select(-number) %>%
  rename(roster_id = hb0, age = hb1, weight = hb2, height = hb3) %>%
  mutate(female = FALSE)
```

# Join — more dplyr

Mutating joins combine variables from the two data.frames:

`inner_join()` return all rows from x where there are matching values in y, and all columns from x and y.
If there are multiple matches between x and y, all combination of the matches are returned.

`left_join()` return all rows from x, and all columns from x and y. Rows in x with no match in y will have NA values in the new columns. If multiple matches between x and y, all combinations of the matches are returned.

`right_join()` return all rows from y, and all columns from x and y. Rows in y with no match in x will have NA values in the new columns. If multiple matches between x and y, all combinations of the matches are returned.

`full_join()` return all rows and all columns from both x and y. Where there are not matching values, returns NA for the one missing.

Filtering joins keep cases from the left-hand data.frame:

`semi_join()` return all rows from x where there are matching values in y, keeping just columns from x. A semi join differs from an inner join because an inner join will return one row of x for each matching row of y, where a semi join will never duplicate rows of x.

`anti_join()` return all rows from x where there are not matching values in y, keeping just columns from x.

# Putting it all together!

```
base <- bind_rows(female, male) %>%  # Combine male and female
  inner_join(educ) %>%  # could also use left here
  inner_join(hh) # note R figures out what to merge on
```

```
## Joining, by = c("hhid", "roster_id")
```

```
## Joining, by = "hhid"
```

What would happen if we used `right_join`?

# Nicer names and tidy up!!

```
base <- base %>%
  rename(state = hv024,
         urban_rural = hv025,
         type_place = hv026,
         wealth_index = hv270) %>%
  select(hhid:educ, state, urban_rural,
         type_place, wealth_index) %>%
  mutate(age = as.numeric(age),
         weight = as.numeric(weight),
         height = as.numeric(height),
         educ = as.numeric(educ))
```

You can also remove the objects we no longer need

```
rm(educ, female, hh, male, nfhs)
```

# Exploratory Data Analysis

# EDA Process

- Is your data **tidy**?

- Use both non-graphical and graphical EDA

(cont.)

- Univariate and multivariate
  - categorical (i.e. **factor**)
    - univariate: count, proportion, percentage
    - multivariate: cross-tabluation if multiple categorical variables
  - quantitative
    - univariate: central tendancy, dispersion, skewness, kurtosis, etc
    - multivariate: correlation
  - categorical and quantitative: multivariate
    - distribution or summary of quantitative by level of categorical variable

# More tidyverse!!

- focus on specific groups
  - `group_by()` will do subset calculations by factor levels
- reduce/summarise data: get a descriptive statistic
  - `summarise()` to create a descriptive statistic using `sum()`, `mean()`, `n()`
- `ggplot2` to graph data
  - A powerful graphing tool with many, many options
  - gg stands for “grammar of graphics”

# Simple first!

```
# Get descriptitve statistics
summary(base)
```

```
##      hhid            roster_id             age           weight
##  Length:228426      Length:228426      Min.   :15.0   Min.   : 151
##  Class :character   Class :character   1st Qu.:21.0   1st Qu.: 447
##  Mode  :character   Mode  :character   Median :29.0   Median : 521
##                                        Mean   :29.8   Mean   :1974
##                                        3rd Qu.:38.0   3rd Qu.: 653
##                                        Max.   :54.0   Max.   :9999
##      height       female             educ           state
##  Min.   : 800   Mode :logical   Min.   : 0.000   Length:228426
##  1st Qu.:1514   FALSE:89834     1st Qu.: 0.000   Class :character
##  Median :1580   TRUE :138592    Median : 8.000   Mode  :character
##  Mean   :2867                   Mean   : 7.032
##  3rd Qu.:1681                   3rd Qu.:10.000
##  Max.   :9999                   Max.   :99.000
##  urban_rural         type_place        wealth_index
##  Length:228426      Length:228426      Length:228426
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

# ehhh?

What is wrong here?

# Back to the data

- Manual says: “height in centimeters (1 decimal)” and “weight in kilograms (1 decimal)”,
- but what is up with the 99 and 9999?

```
base <- base %>%
  mutate(
    weight = case_when(
      weight <= 9000 ~ weight / 10,
      weight  > 9000 ~ NA_real_
    ),
    height = case_when(
      height <= 9000 ~ height / 10,
      height  > 9000 ~ NA_real_
    ),
    educ = case_when(
      educ <= 90 ~ educ,
      educ  > 90 ~ NA_real_
    )
  )
```

# Try again!

```
# Get descriptitve statistics
summary(base)
```

```
##      hhid            roster_id             age           weight      
##  Length:228426      Length:228426      Min.   :15.0   Min.   : 15.10  
##  Class :character   Class :character   1st Qu.:21.0   1st Qu.: 43.50  
##  Mode  :character   Mode  :character   Median :29.0   Median : 49.70  
##                                        Mean   :29.8   Mean   : 51.52  
##                                        3rd Qu.:38.0   3rd Qu.: 57.60  
##                                        Max.   :54.0   Max.   :173.00  
##                                                       NA's   :35131   
##      height        female             educ            state          
##  Min.   : 80.0   Mode :logical   Min.   : 0.000   Length:228426      
##  1st Qu.:150.3   FALSE:89834     1st Qu.: 0.000   Class :character   
##  Median :155.8   TRUE :138592    Median : 8.000   Mode  :character   
##  Mean   :156.7                   Mean   : 6.887                      
##  3rd Qu.:162.6                   3rd Qu.:10.000                      
##  Max.   :199.3                   Max.   :23.000                      
##  NA's   :35216                   NA's   :359                         
##  urban_rural         type_place        wealth_index      
##  Length:228426      Length:228426      Length:228426      
##  Class :character   Class :character   Class :character   
##  Mode  :character   Mode  :character   Mode  :character   
##                                                           
##                                                           
```

# Univariate non-graphical

## Categorical data - tabulation

```
# Make table to show quantitative data by place
base %>%                      # %>% Pipes data to group_by() function
  group_by(type_place) %>%    # group_by() organises data by place
  summarise(count = n())      # summarise() reduces to descriptive stat
```

```
## # A tibble: 4 x 2
##   type_place           count
##   <chr>                <int>
## 1 capital, large city  55440
## 2 countryside         117474
## 3 small city           16346
## 4 town                 39166
```

## So common that:

```
base %>%                      # %>% Pipes data to group_by() function
  group_by(type_place) %>%    # group_by() organises data by place
  count()                     # What it says on the tin!
```

```
## # A tibble: 4 x 2
## # Groups:   type_place [4]
##   type_place              n
##   <chr>               <int>
## 1 capital, large city  55440
## 2 countryside         117474
## 3 small city           16346
## 4 town                 39166
```

# Calculate more descriptive statistics

```
# Make table to show quantitative data by type of place
base %>%                      # %>% Pipes data to group_by() function
  group_by(type_place) %>%  # group_by() organises data by place
  summarise(count = n(),    # count is new variable
    # percent is a new variable, sum() and nrow() are functions
    percent = (sum(count) / nrow(base)) * 100,
    mean_age = mean(age),           # mean age by place
    mean_height = mean(height),     # mean height by place
    mean_weight = mean(weight))     # mean weight by place
```

```
## # A tibble: 4 x 6
##   type_place           count percent mean_age mean_height mean_weight
##   <chr>                <int>   <dbl>    <dbl>       <dbl>       <dbl>
## 1 capital, large city  55440   24.3      30.1          NA          NA
## 2 countryside         117474   51.4      29.6          NA          NA
## 3 small city           16346    7.16     30.0          NA          NA
## 4 town                 39166   17.1      29.9          NA          NA
```

# What is up with height and weight?

```
# Make table to show quantitative data by type of place
base %>%
  group_by(type_place) %>%
  summarise(count = n(),
    percent = (sum(count) / nrow(base)) * 100,
    mean_age = mean(age),
    mean_height = mean(height, na.rm = TRUE ),
    mean_weight = mean(weight, na.rm = TRUE))
```

```
## # A tibble: 4 x 6
##   type_place           count percent mean_age mean_height mean_weight
##   <chr>                <int>   <dbl>    <dbl>       <dbl>       <dbl>
## 1 capital, large city  55440   24.3      30.1        158.        55.5
## 2 countryside         117474   51.4      29.6        156.        49.1
## 3 small city           16346    7.16     30.0        157.        53.7
## 4 town                 39166   17.1      29.9        157.        53.1
```

# If female instead

```
# Make table to show quantitative data by female
base %>%
  group_by(female) %>%     # group_by() organises data by female
  summarise(count = n(),
    percent = (sum(count) / nrow(base)) * 100,
    mean_age = mean(age),
    mean_height = mean(height, na.rm = TRUE ),
    mean_weight = mean(weight, na.rm = TRUE))
```

```
## # A tibble: 2 x 6
##   female  count percent mean_age mean_height mean_weight
##   <lgl>   <int>   <dbl>    <dbl>       <dbl>       <dbl>
## 1 FALSE   89834    39.3     31.0        164.        56.3
## 2 TRUE   138592    60.7     29.0        152.        48.7
```

# If both place and female

```
# Make table to show quantitative data by place and female
base %>%
  group_by(type_place, female) %>% # what happens if switch order??
  summarise(count = n(),
    percent = (sum(count) / nrow(base)) * 100,
    mean_age = mean(age),
    mean_height = mean(height, na.rm = TRUE ),
    mean_weight = mean(weight, na.rm = TRUE))
```

```
## # A tibble: 8 x 7
## # Groups:   type_place [?]
##   type_place          female count percent mean_age mean_height mean_weight
##   <chr>               <lgl>  <int>   <dbl>    <dbl>       <dbl>       <dbl>
## 1 capital, large city FALSE  25591   11.2      30.8        166.        59.4
## 2 capital, large city TRUE   29849   13.1      29.5        153.        52.4
## 3 countryside         FALSE  42804   18.7      31.0        164.        53.8
## 4 countryside         TRUE   74670   32.7      28.8        152.        46.6
## 5 small city          FALSE   6010    2.63     31.1        165.        58.3
## 6 small city          TRUE   10336    4.52     29.4        153.        51.2
## 7 town                FALSE  15429    6.75     31.2        164.        57.7
## 8 town                TRUE   23737   10.4      29.1        152.        50.4
```

# Graphing basics

## Terminology

- Glyph - observation or case of the data set
  - a mark on the graph
  - can represent multiple attributes (variables) of an observation
  - each attribute of glyph can be a graphical element
- Graphic - type of graph
  - how the glyphs are arranged
- Mapping - variable to graph attribute

# Basic graph types:

Distributions - histograms or pdf

Bar chart - summary of a variable, like mean or sum

Scatter plot - relationship between variables

# ggplot2 basics

ggplot is more complicated than base plot

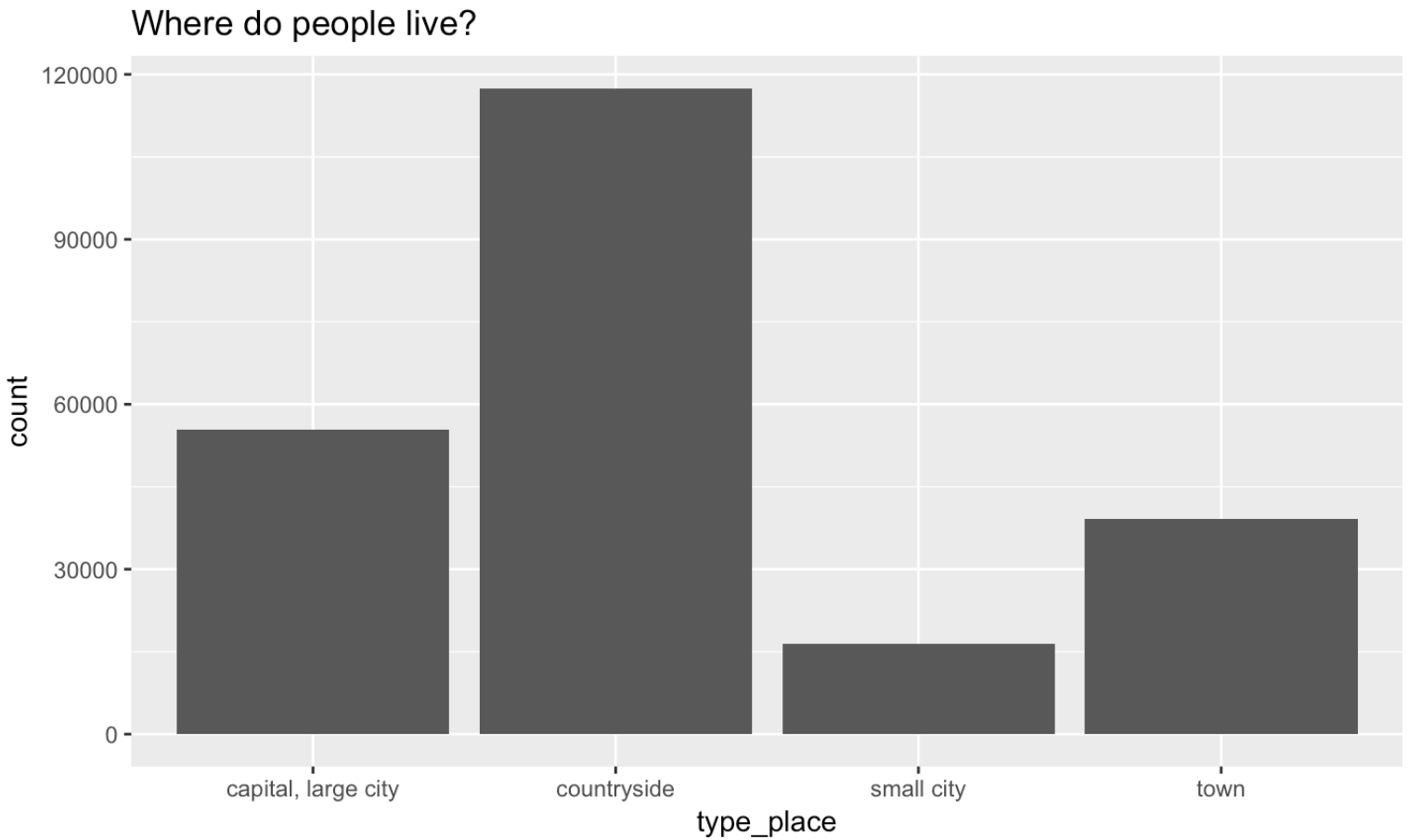
For quick graphs in base R can use `qplot()`
- I won't spend time on it, but you can try it

1. Specify a **frame**
    - **frame** establishes a coordinate system
    - must define what **asethetics** use in frame
        - **asethetic** is an attribure of a glyph, e.g. x, y or other variable
    - `ggplot()` sets up frame
2. Specify geometric object
    - what type of graph: bar, scatter, line, etc

ggplot uses a layering approach
- will add layers to make the graph more detailed
- use “+” to add layers

# Univariate graphical – Categorical

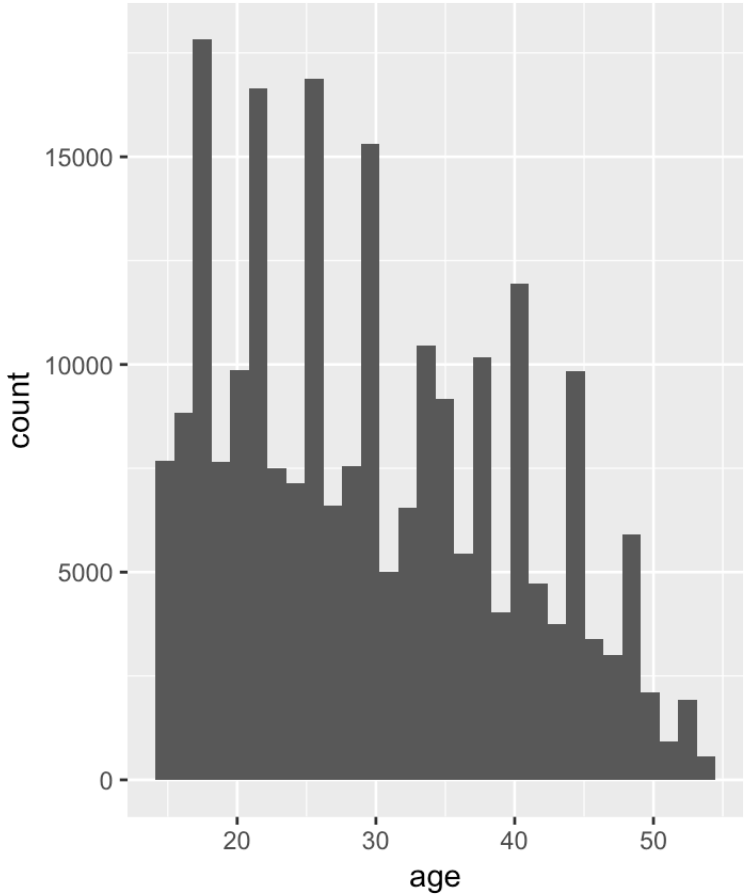
```
# Use ggplot2 to make a bar plot of place
base %>%
  group_by(type_place) %>%  # group_by() groups all data by place
  # aes() sets what will be on the x-axis, use + to add layer
  ggplot(aes(x = type_place)) +
    # geom_bar() determines how data will be arranged, count is default
    geom_bar() +
    ggtitle("Where do people live?") # add title layer
```

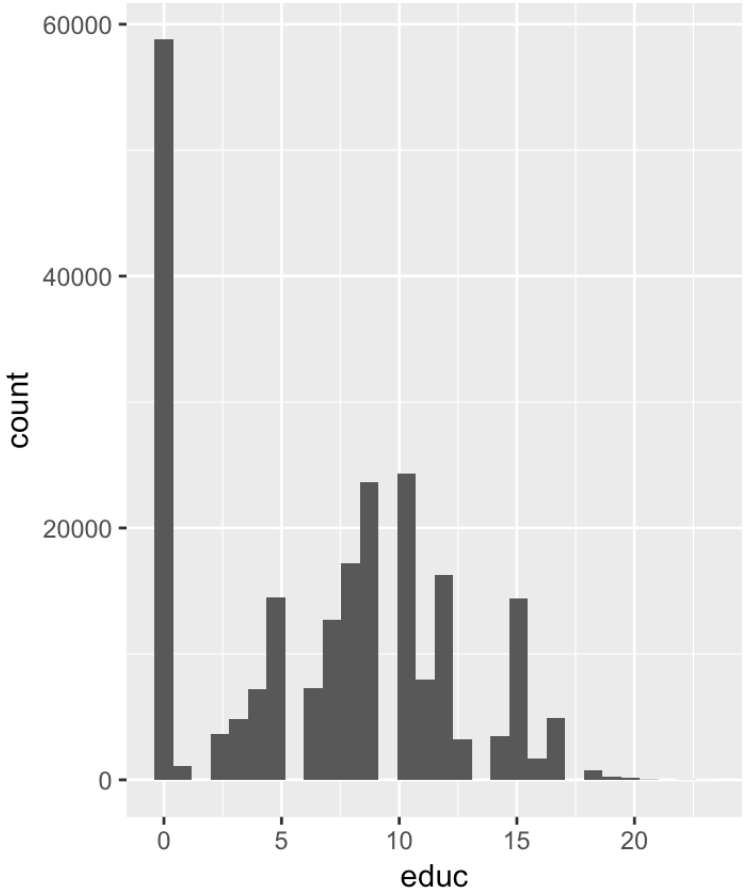

Where do people live?
120000
90000
60000
30000
0
count
capital, large city
countryside
small city
town
type_place

# Histograms in ggplot2

```
# Loaded gridExtra so can plot more than one graph with grid.arrange
grid.arrange(
# First graph
  ggplot(base, aes(x = age)) +
    geom_histogram(),

# Second graph
  ggplot(base, aes(x = educ)) +
    geom_histogram(),

# Specify number of columns, like using par(mfrow = c(1, 2))
  ncol = 2
)
```

```
## Warning: Removed 359 rows containing non-finite values (stat_bin).
```

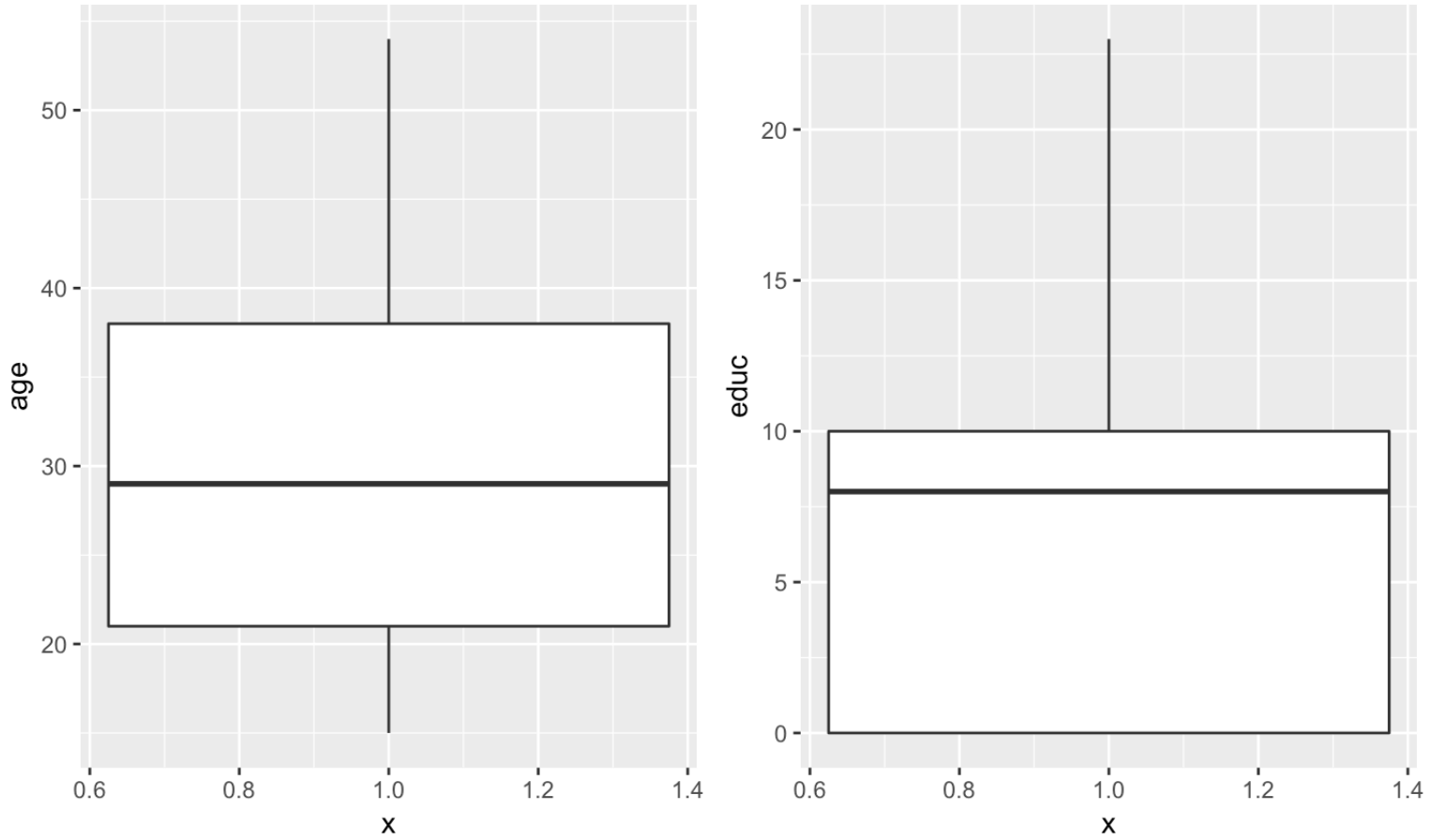


# Boxplots of quantitative data in ggplot2

```
# If you want to make a boxplot of a single continuous variable you
# have to set x = 1 because the boxplot does not have a width
# associated with it so gives alignment
grid.arrange(

  ggplot(base, aes(x = 1, y = age)) +
    geom_boxplot(),

  ggplot(base, aes(x = 1, y = educ)) +
    geom_boxplot(),

  ncol = 2
)
```

```
## Warning: Removed 359 rows containing non-finite values (stat_boxplot).
```



# Multivariate non-graphical – Categorical

Use cross-tabs for counts/proportion/percent

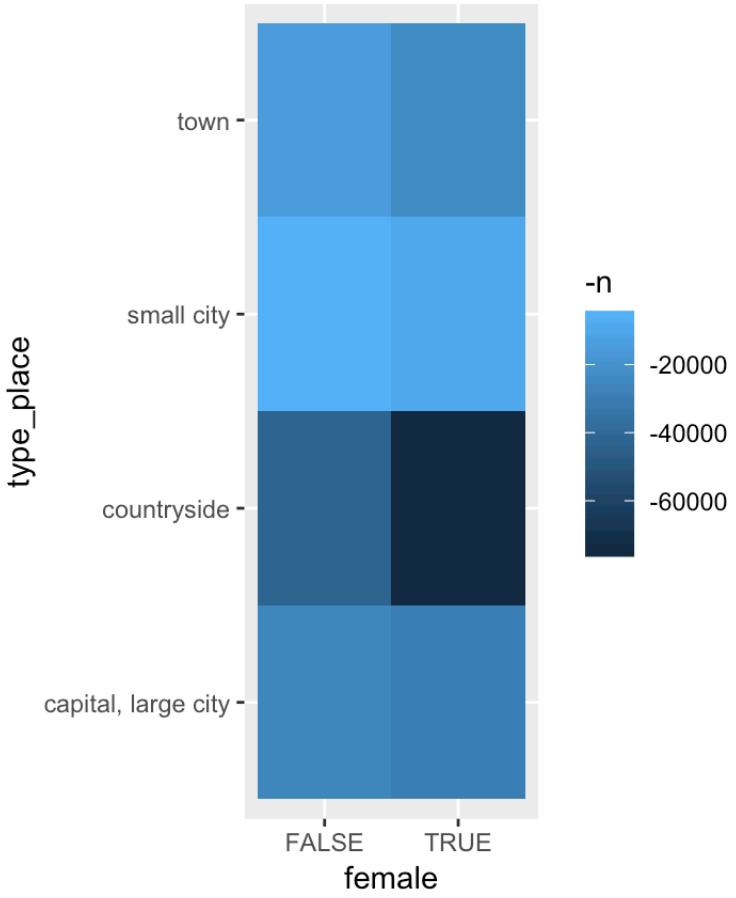
```
# addmargins and xtabs are base commands that create
# a crosstab with sum of row and column
addmargins(xtabs( ~ type_place + female,  data = base))
```

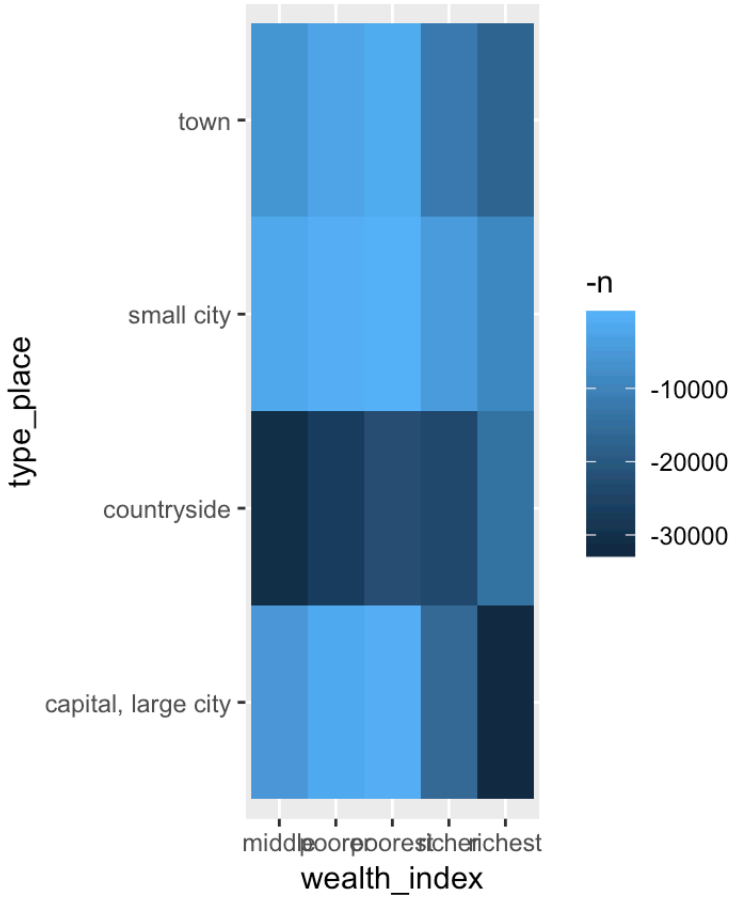
```
##                       female
## type_place              FALSE   TRUE    Sum
##   capital, large city  25591  29849  55440
##   countryside          42804  74670 117474
##   small city            6010  10336  16346
##   town                 15429  23737  39166
##   Sum                  89834 138592 228426
```

# Multivariate graphical – Categorical

Not many options for multivariate EDA with categorical variables

```
grid.arrange(

base %>%
  count(female, type_place) %>%
  ggplot(aes(x = female, y = type_place)) +
    geom_tile(aes(fill = -n)),
   # geom_tile is used for two categorical variables
   # n tells which attribute to base color on
   # -n the "-" says to go in ascending order

base %>%
  count(wealth_index, type_place) %>%
  ggplot(aes(x = wealth_index, y = type_place)) +
    geom_tile(aes(fill = -n)),

ncol = 2
)
```


town
small city
countryside
capital, large city
type_place
FALSE
TRUE
female
-n
-20000
-40000
-60000


town
small city
countryside
capital, large city
type_place
middle
poorer
poorest
richer
richest
wealth_index
-n
-10000
-20000
-30000

# Multivariate non-graphical – Quantitative

The standard measure between quantitative variables is correlation

```
cor(base$weight, base$height, use = "complete.obs")
```

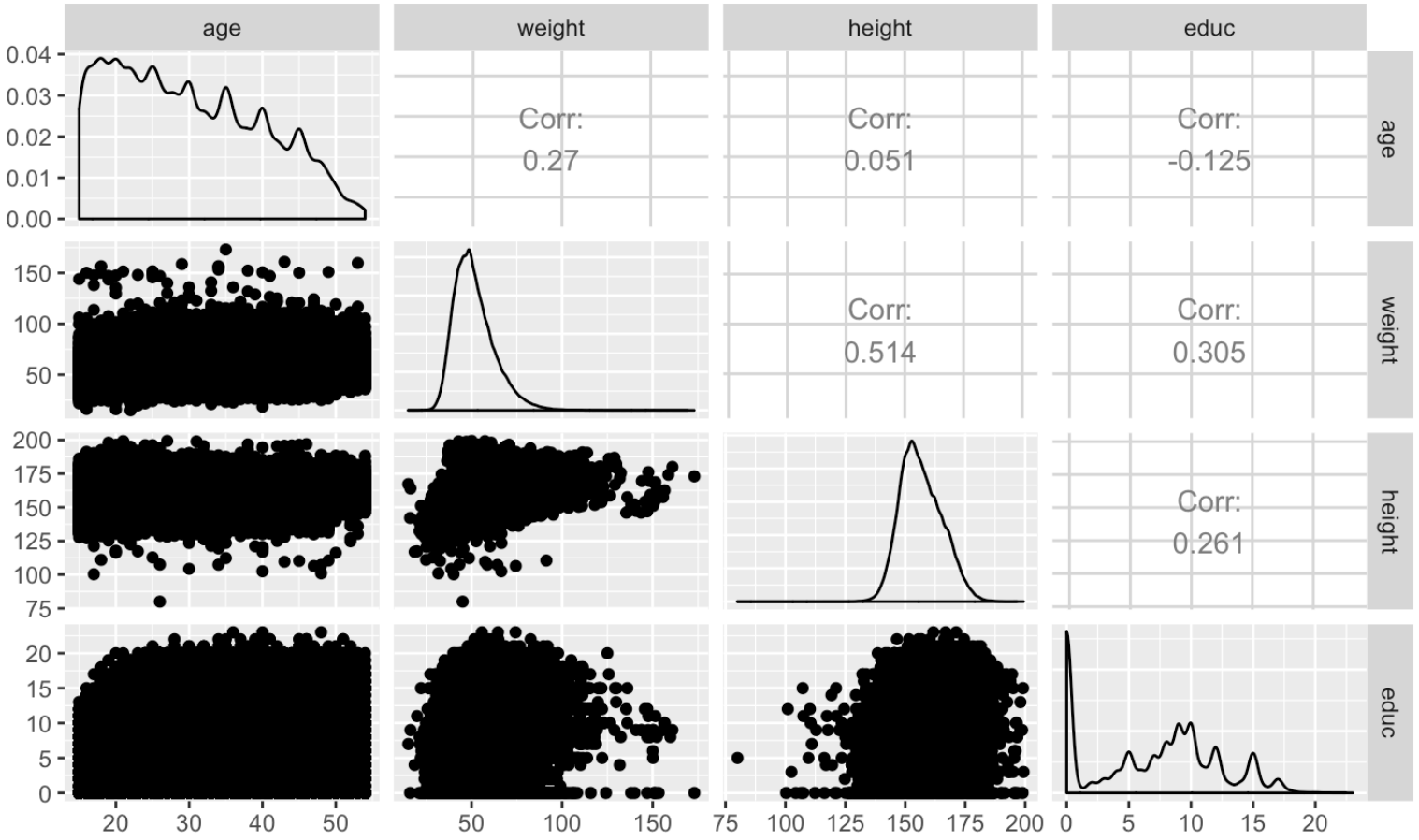
```
## [1] 0.5135399
```

# Multivariate graphical – Quantitative

There are many options when using quantitative variables

Begin with a correlation graph of all variables

```
# Load GGally package to get a custom graph of
# correlation not in ggplot2
ggpairs(base, columns = c("age", "weight", "height", "educ"))
```


age
weight
height
educ
Corr:
0.27
Corr:
0.051
Corr:
-0.125
Corr:
0.514
Corr:
0.305
Corr:
0.261
0.04
0.03
0.02
0.01
0.00
150
100
50
200
175
150
125
100
75
20
15
10
5
0
20
30
40
50
50
100
150
75
100
125
150
175
200
0
5
10
15
20

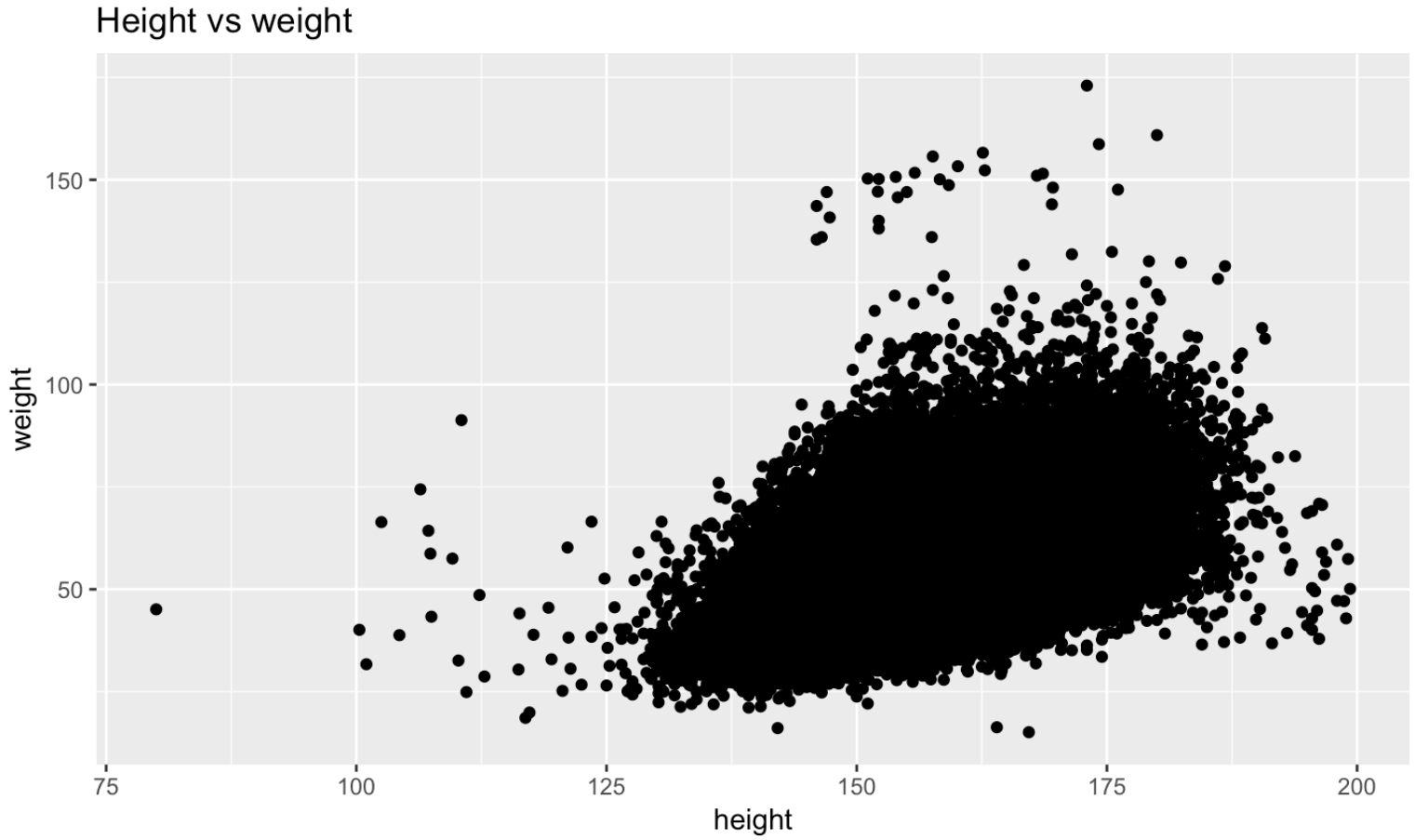
# Many options for quantitative variables

## Usually some form of a scatter plot

- Can use size, shape, color to identify additional variables
- Begin with standard scatter plot

```
ggplot(base, aes(x = height, y = weight)) +
  geom_point() +
  ggtitle("Height vs weight")
```

```
## Warning: Removed 35251 rows containing missing values (geom_point).
```



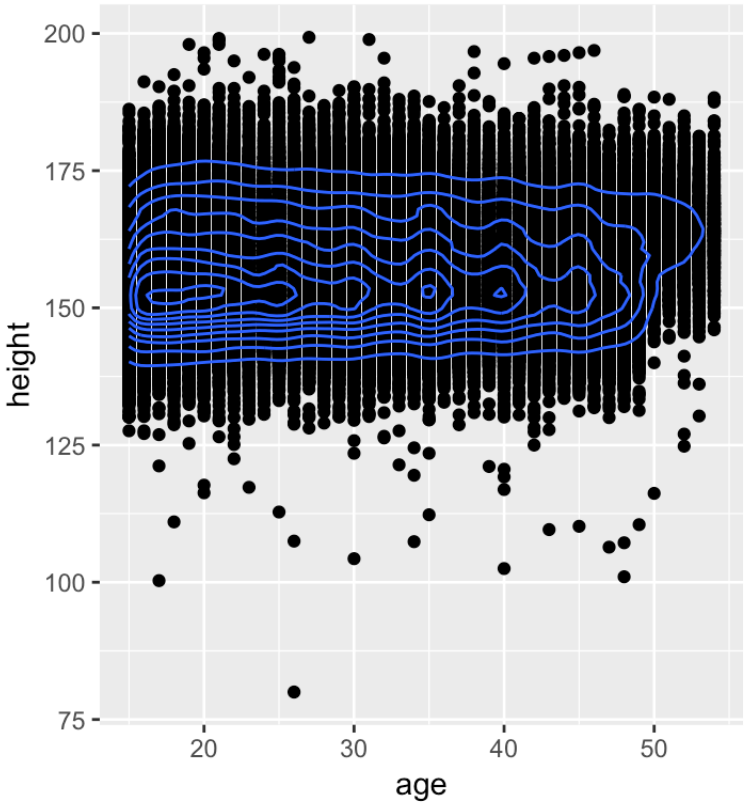
# Other dimensions of the data

## Illustrate density of observations with contour or heat

```
grid.arrange(

ggplot(base, aes(x = age, y = height)) +
  geom_point() +
  stat_density2d() +
  ggtitle("Many observations at young ages",
          subtitle = "Contour"),

ggplot(base, aes(x = age, y = height)) +
  geom_point() +
  stat_density2d(aes(fill = ..density..), geom = "raster",
                 contour = FALSE) +
  ggtitle("Heat map",
          subtitle = "Light color indicates more observations!"),

ncol = 2
)
```

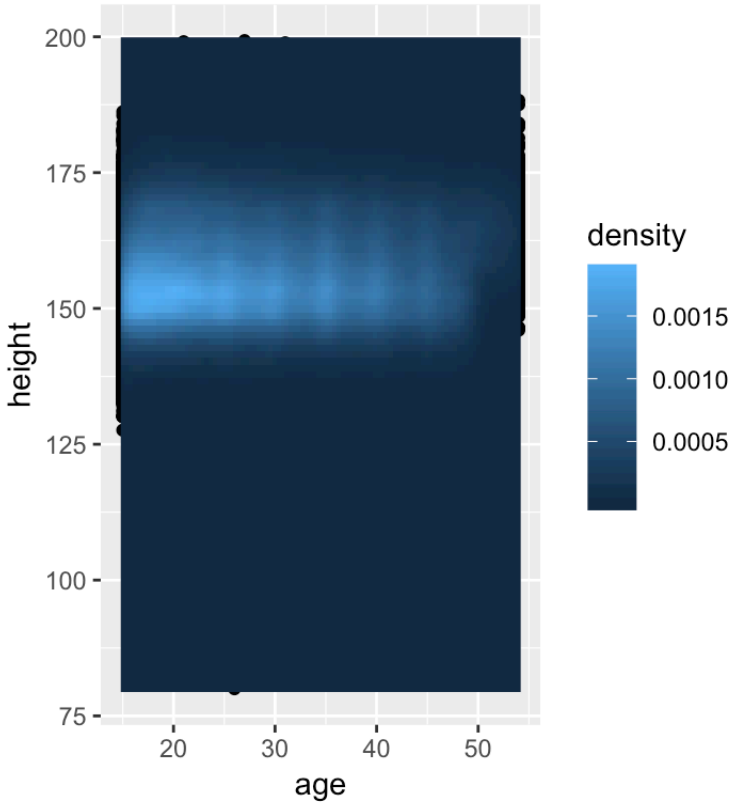
Many observations at young ages

Contour




Heat map

Light color indicates more observations!



# 9. Multivariate non-graphical – Categorical and Quantitative

There are many options depending on what you want to learn

- These options usually take the form of a **reduction** by level
  - **Reduction** is of the quantitative variable
  - Reduction is done by level of the categorical variable
- Table of mean and count of age and expenditures (reductions) by ethnicity (level)

```
base %>%
  group_by(wealth_index) %>%
  summarise(age_mean = mean(age),
            height_mean = mean(height, na.rm = TRUE),
            n = n(),
            percent = n / nrow(base) * 100)
```

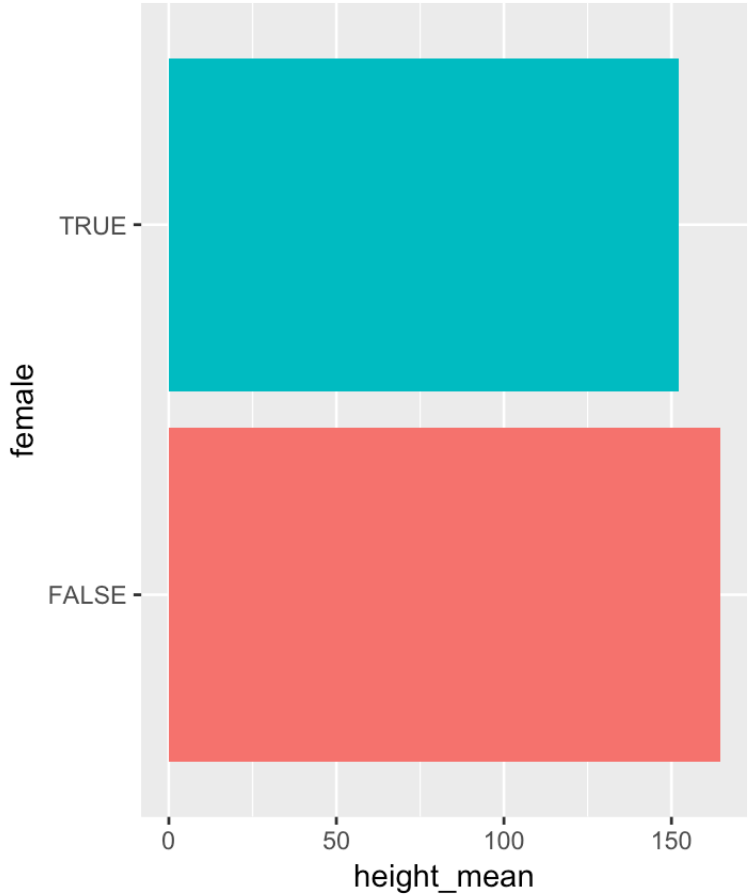
```
## # A tibble: 5 x 5
##   wealth_index age_mean height_mean     n percent
##   <chr>           <dbl>       <dbl> <int>   <dbl>
## 1 middle           29.3        156. 43768    19.2
## 2 poorer           29.4        155. 31766    13.9
## 3 poorest          29.5        154. 24119    10.6
## 4 richer           29.5        157. 56532    24.7
## 5 richest          30.6        158. 72241    31.6
```

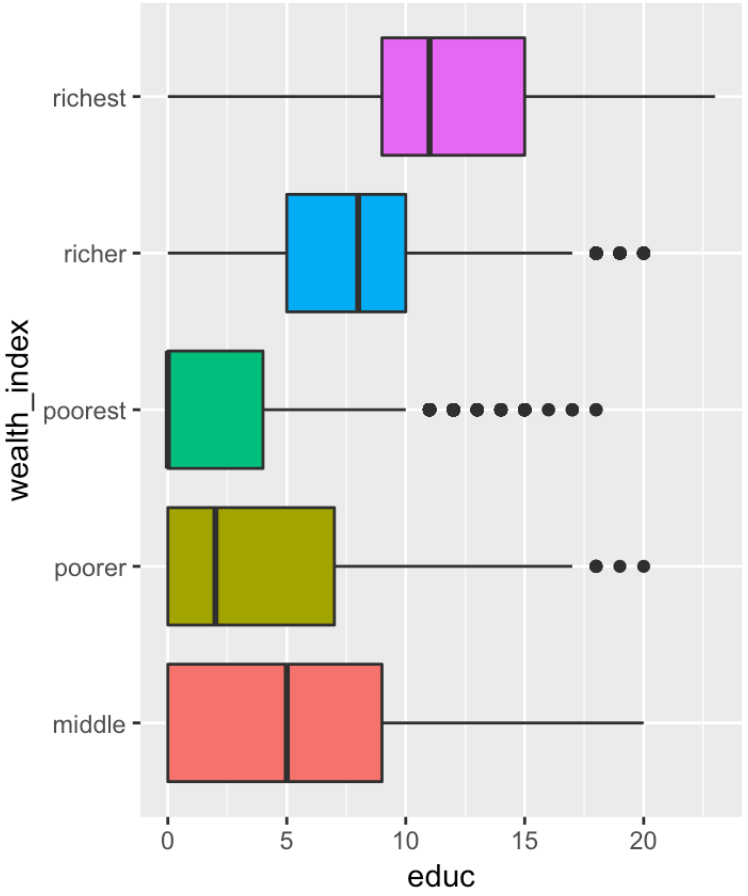
# Multivariate graphical

## Categorical and Quantitative

```
# Bar and boxplot
grid.arrange(

base %>%
  group_by(female) %>%
  summarise(height_mean = mean(height, na.rm = TRUE)) %>%
  ggplot(aes(x = female, y = height_mean, fill = female)) +
    geom_bar(stat = "identity") +   # stat="identity" means use the y value as a column
    coord_flip() +   # coord_flip() changes bars from verticle to horizontal
    guides(fill = FALSE, ylab = FALSE), # this removes the legend

base %>%
  group_by(wealth_index) %>%
  ggplot(aes(x = wealth_index, y = educ, fill = wealth_index)) +
    geom_boxplot() +
    coord_flip() +
    guides(fill = FALSE, ylab = FALSE),

ncol = 2
)
```

```
## Warning: Removed 359 rows containing non-finite values (stat_boxplot).
```

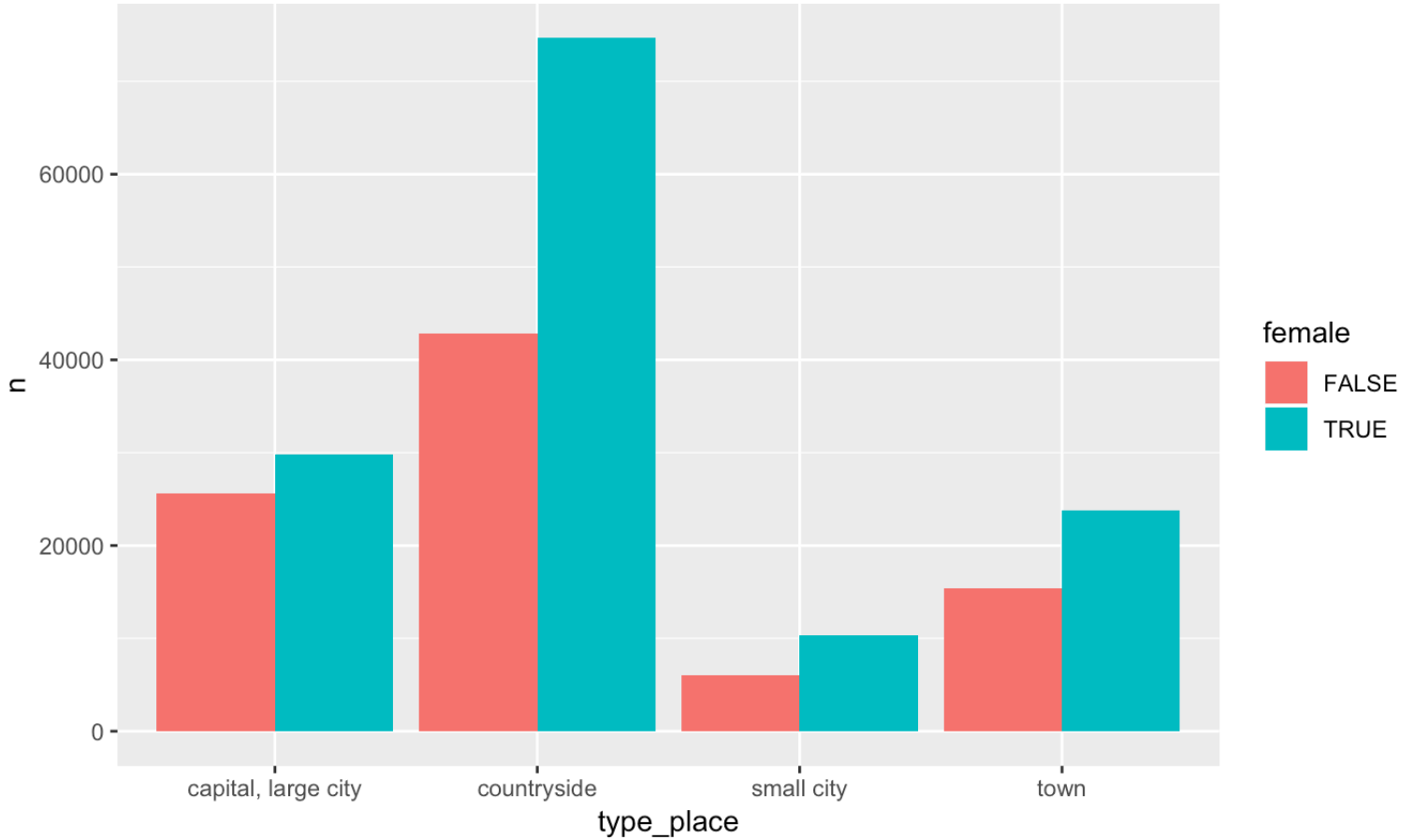


# Two categorical and one quantitative

## Clustered bar graph

```
# Basic graph with place by color
base %>%
  count(type_place, female) %>%
  ggplot(aes(x = type_place, y = n, fill = female)) +
    geom_bar(stat = "identity", position = "dodge")
```


60000
40000
20000
0
n
capital, large city
countryside
small city
town
type_place
female
FALSE
TRUE

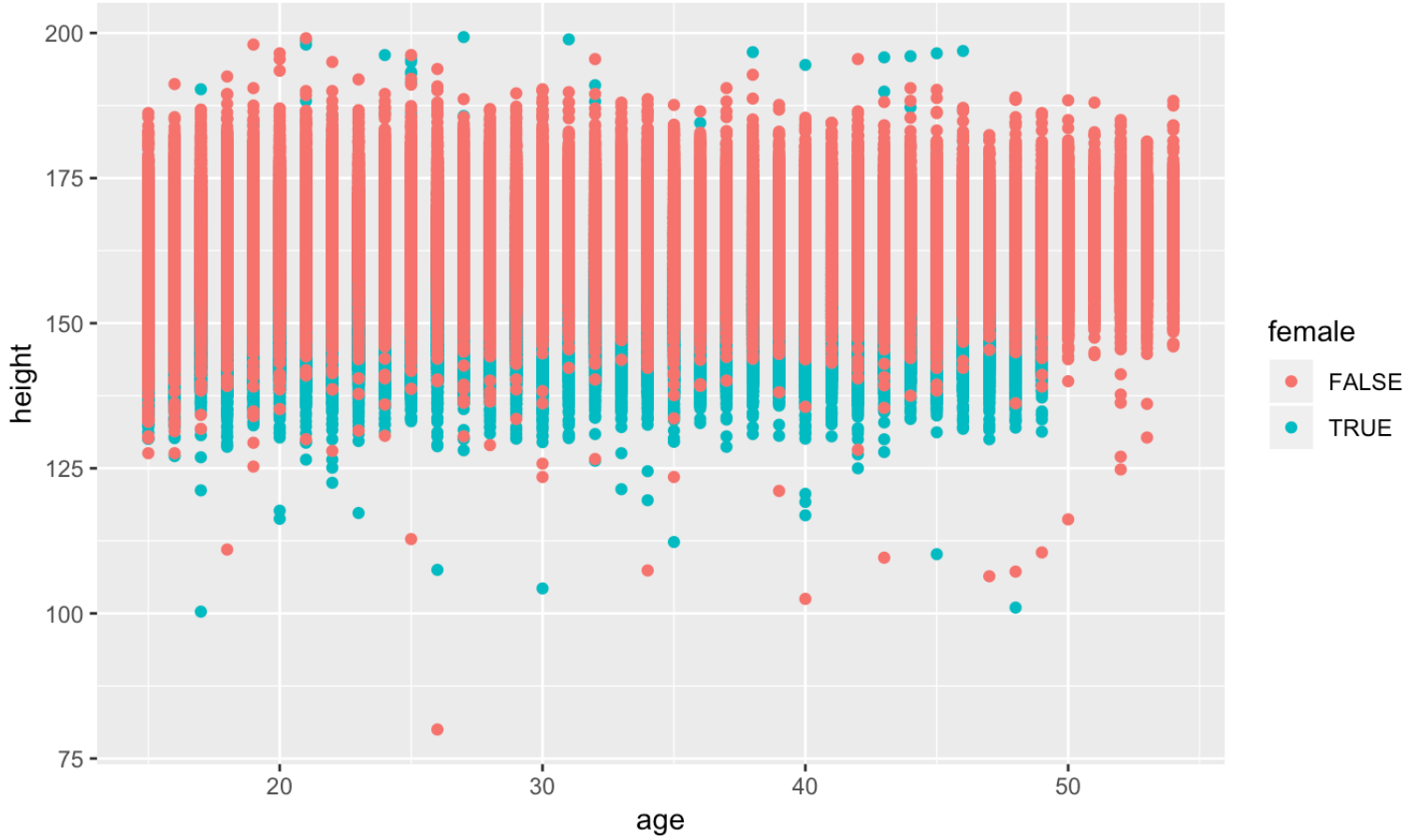
# More than one categorical and one quantitative

Can use characteristics like color, size and shape

- Two quantitative and one categorical

```
# Basic graph with ethnicity by color
ggplot(base, aes(x = age, y = height, color = female)) +
  geom_point()
```

```
## Warning: Removed 35216 rows containing missing values (geom_point).
```

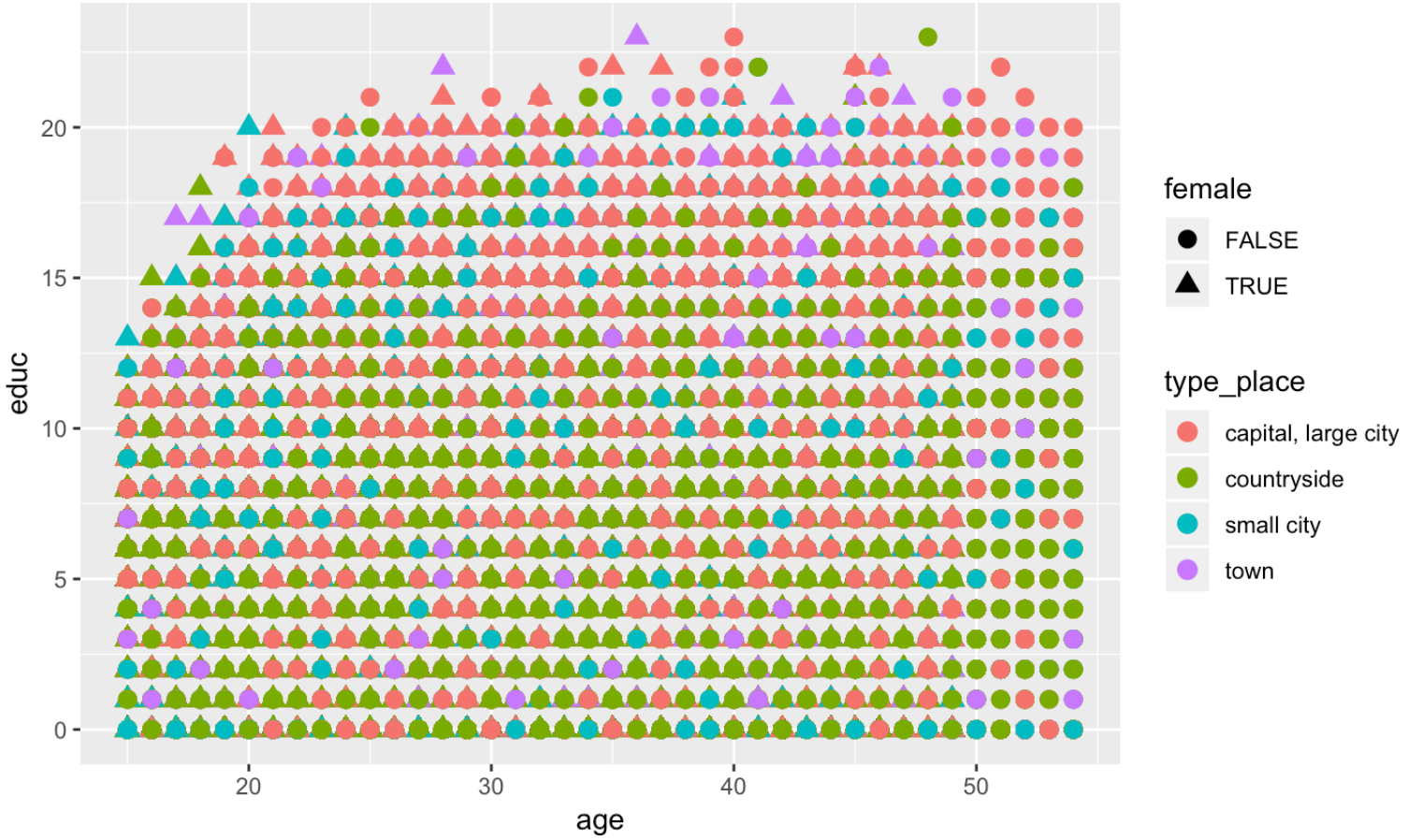


# More than one categorical and one quantitative

- Two quantitative and two categorical
- What is wrong with this graph?

```
ggplot(base, aes(x = age, y = educ,
                 color = type_place,
                 shape = female)) +
  geom_point(size = 3)
```

```
## Warning: Removed 359 rows containing missing values (geom_point).
```



# EDA Summary

EDA process is to explore using:

- non-graphical and graphical analysis

- Univariate and multivariate

## Using tidyverse versus base

- I like to use a mix of tidyverse and base
    - Base code and graphing for quick EDA
    - Tidyverse for more detailed code and graphs
- Tidyverse often eaiser to read because:
    - function names make sense
    - piping/chaining breaks up steps
    - able to have a cleaner environment

# Some Basic Statistics

# Central Limit Theorem

The sampling distribution of the mean of a random sample drawn from any population is approximately normal for a sufficiendy large sample size. The larger the sample size, the more closely the sampling distribution of $\bar{X}$ will resemble a normal distribution.

Remember: independent of the underlying distribution of $X$!

The distribution of means across repeated samples will be normal with a mean equal to the population mean and a standard deviation equal to the population standard deviation divided by the square root of n.

# Sample standard deviation

$$s = \sqrt{\frac{\sum_{i=1}^{n} X_i - \bar{X}}{n-1}}$$

# Hypothesis

Status quo goes in null hypothesis

What you want to “prove” goes in alternative hypothesis

# T test on mean

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

Degress of freedom = $n - 1$

# P values

High P values: your data are likely with a true null.

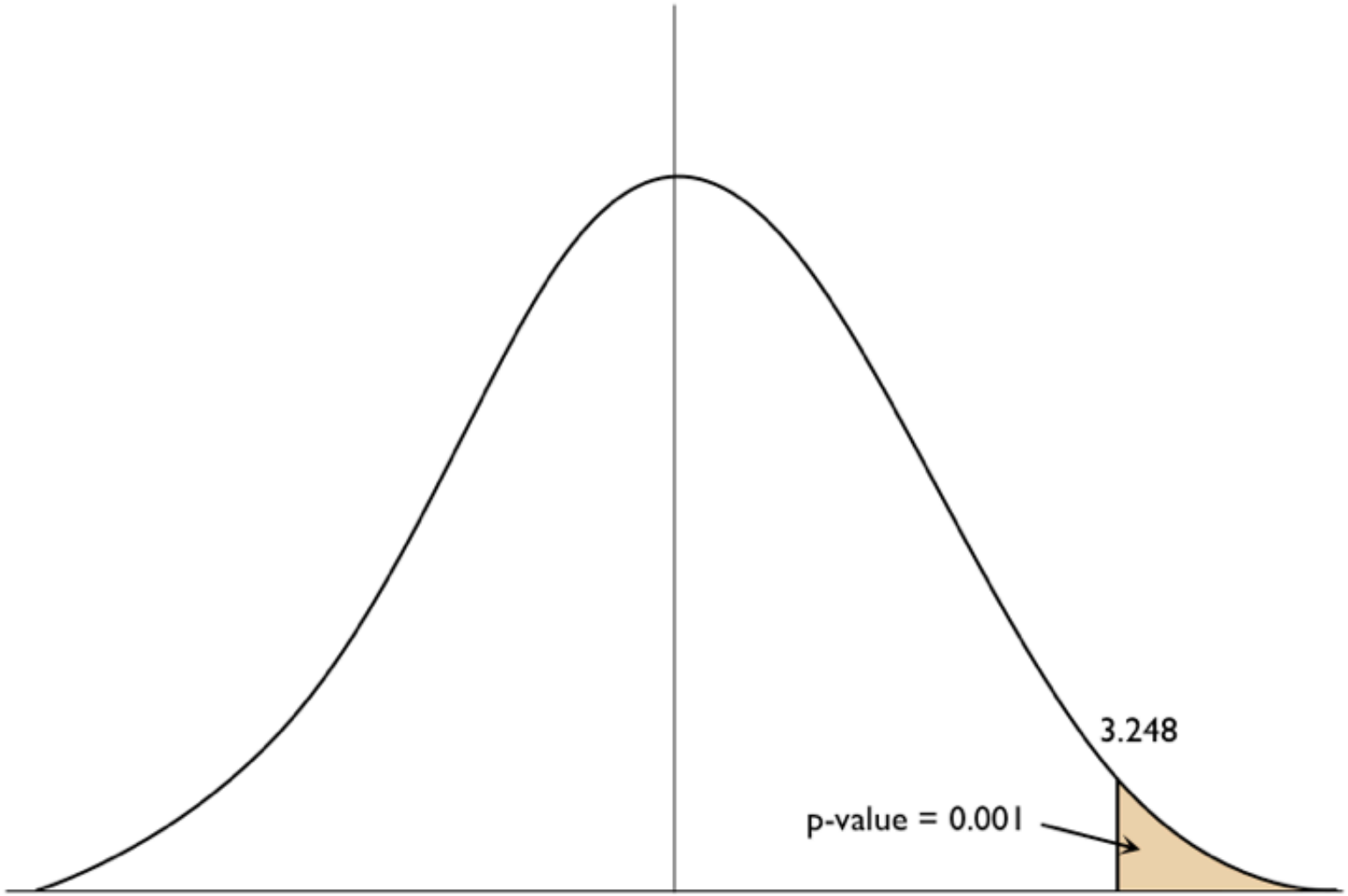
Low P values: your data are unlikely with a true null.

In technical terms: a P value is the probability of obtaining an effect at least as extreme as the one in your sample data, assuming the truth of the null hypothesis.

# More P values

- P values evaluate how well the sample data support the devil's advocate argument that the null hypothesis is true.
- In other words: it measures how compatible your data are with the null hypothesis.
- Or other, other words: How likely is the effect observed in your sample data if the null hypothesis is true?
- Nothing about whether your alternative hypothesis is supported!!

# Testing



# For Next Time

# Reading

- R for Data Science, Part 2
- An Introduction to Statistical Learning
  - Chapter 2
  - Chapter 3, sections 1 and 6.1-6.2
- Wooldridge (if you want the more technical version)
  - Chapter 2