

FE8828 Programming Web Applications in Finance

Week 4: 9. ggplot2: Data Visualization and EDA

Dr. Yang Ye yy@runchee.com

Nanyang Business School

Oct 8, 2020

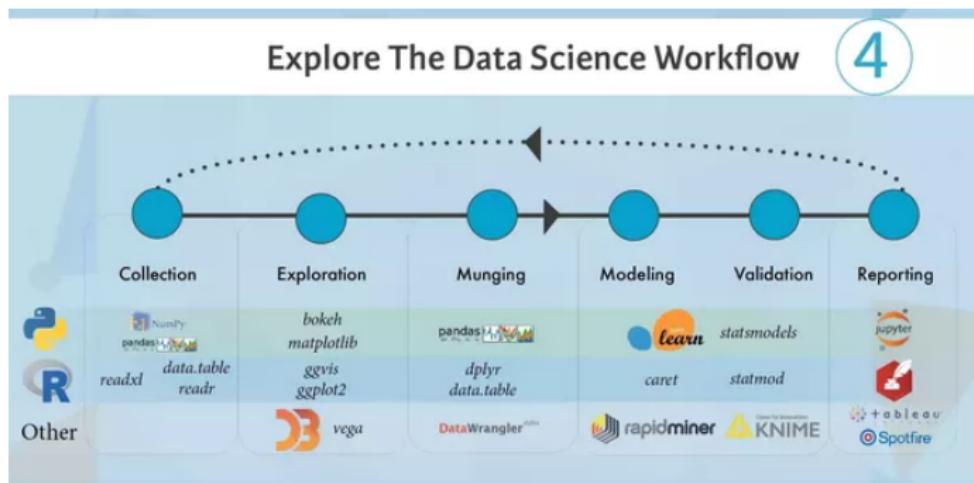
1 Lecture 9: Data Visualization and EDA

Section 1

Lecture 9: Data Visualization and EDA

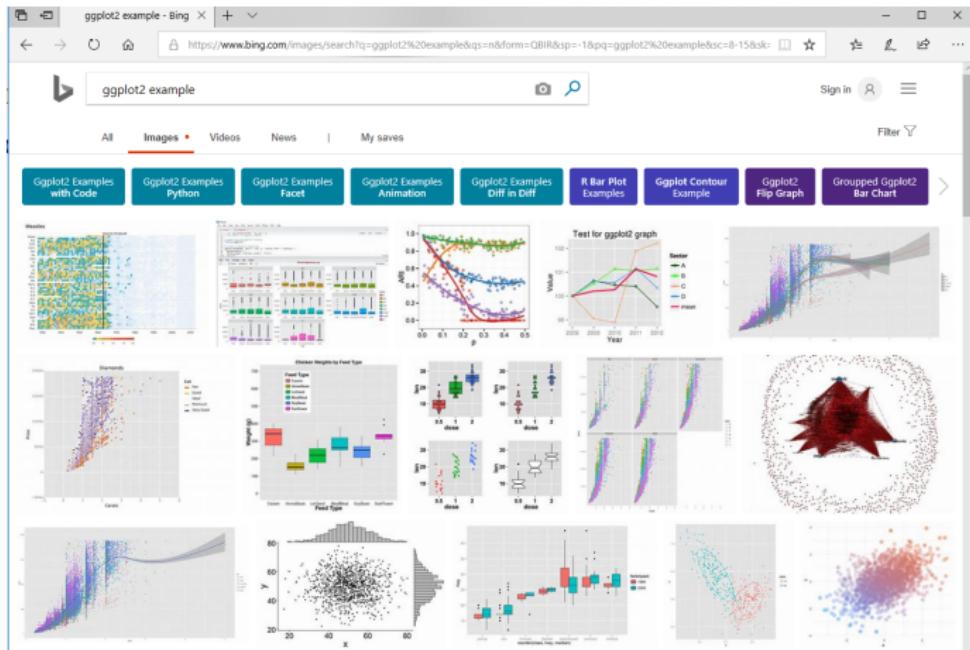
Why do we need ggplot?

Visualizations is part of the data exploration.



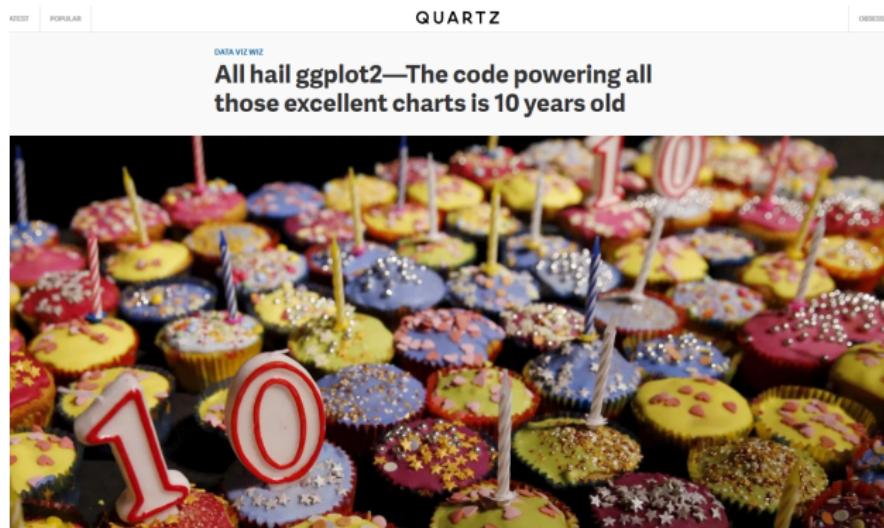
ggplot2

- `library(ggplot2)`, author Hadley Wickham. First release on June 10, 2007.
- gg stands for “Grammar of Graphics”



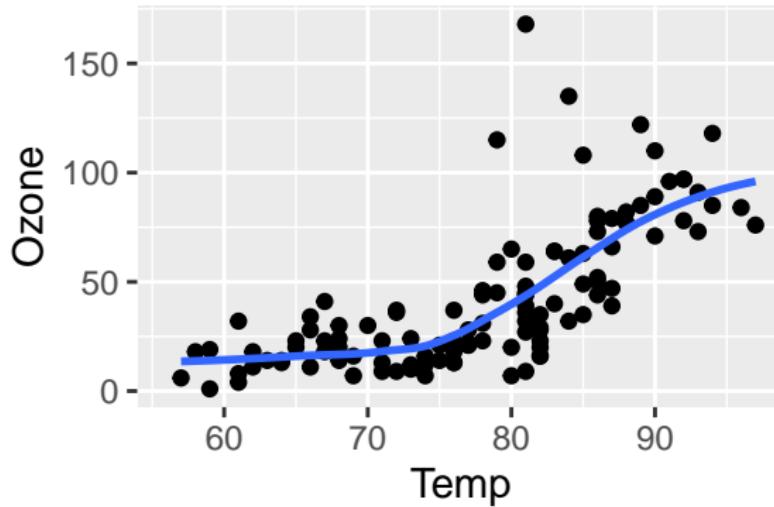
ggplot2 in news

<https://qz.com/1007328/all-hail-ggplot2-the-code-powering-all-those-excellent-charts-is-10-years-old/>



Programming ggplot2

```
# library(ggplot2)
ggplot(airquality, aes(Temp, Ozone)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE)
## `geom_smooth()` using formula 'y ~ x'
## Warning: Removed 37 rows containing non-finite values (stat_smooth).
## Warning: Removed 37 rows containing missing values (geom_point).
```



Syntax of ggplot

- Definition of data + Definitions of layers.

```
# For Single-layer
ggplot(the-data, aes(variable-to-plot)) +
  geom_type-of-graph()

# Or, for multi-layer
ggplot(data = <DATA>, ...) +
  <GEOM_FUNCTION1>(mapping = aes(<MAPPINGS>)) +
  <GEOM_FUNCTION2>(mapping = aes(<MAPPINGS>)) +
  ...
  other formatting functions.
```

Syntax of ggplot: Points to Note

- Put the + sign in the end of the line, not the beginning of the line.

```
# Below would result in error  
ggplot(data = <DATA>, ...)  
+ <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

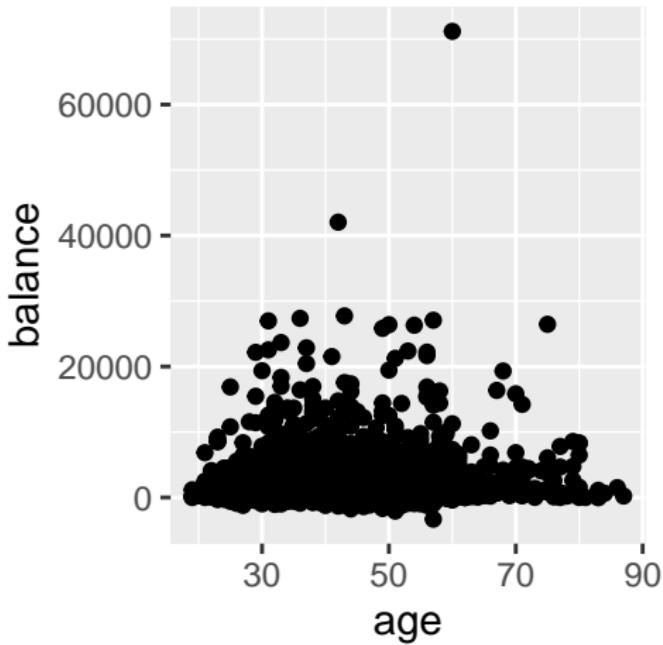
- The definition of data will pass down through the layers. But each layer can have its own data.

```
ggplot(data = d1, ...) +  
  geom_point() # this would get data = d1  
  geom_point(data = d2, ...) # this would get data = d2
```

- `geom_point()` is for points, `geom_line()` is for line, `geom_smooth()` for smoothed line... There are many `geom-etry`.

Simple plot: x and y

```
ggplot(bank, aes(age, balance)) + geom_point()
```



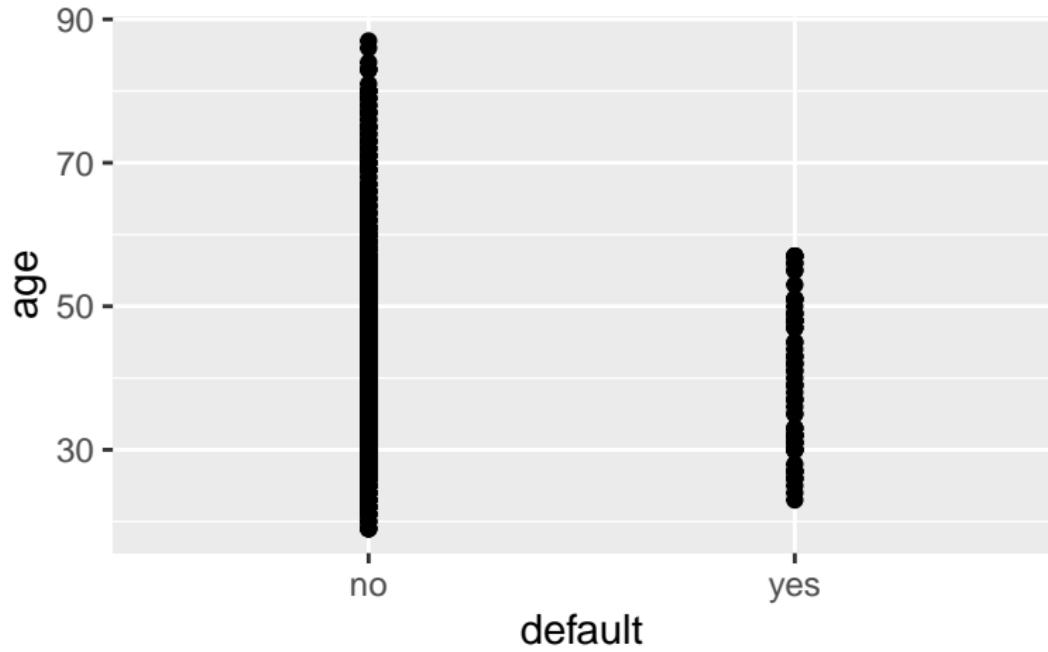
Learning Points for ggplot

- Viz for non-numeric data
- Adjustment
- A few tactics
 - ▶ Multiple geoms
 - ▶ Pass aes down or not
 - ▶ aes and aes_string
- Layers
- Differentiate groups
- Viz for numeric data

For non-numeric data: default and age

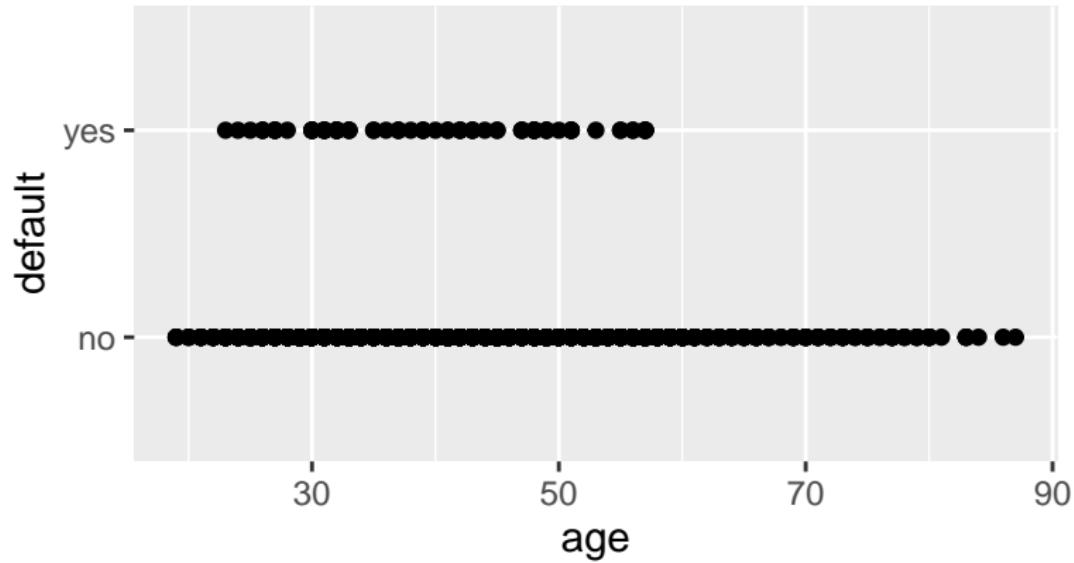
```
ggplot(bank, aes(default, age)) + geom_point()  
ggplot(bank, aes(age, default)) + geom_point()  
ggplot(bank, aes(job, age)) + geom_point()
```

```
ggplot(bank, aes(default, age)) + geom_point()
```



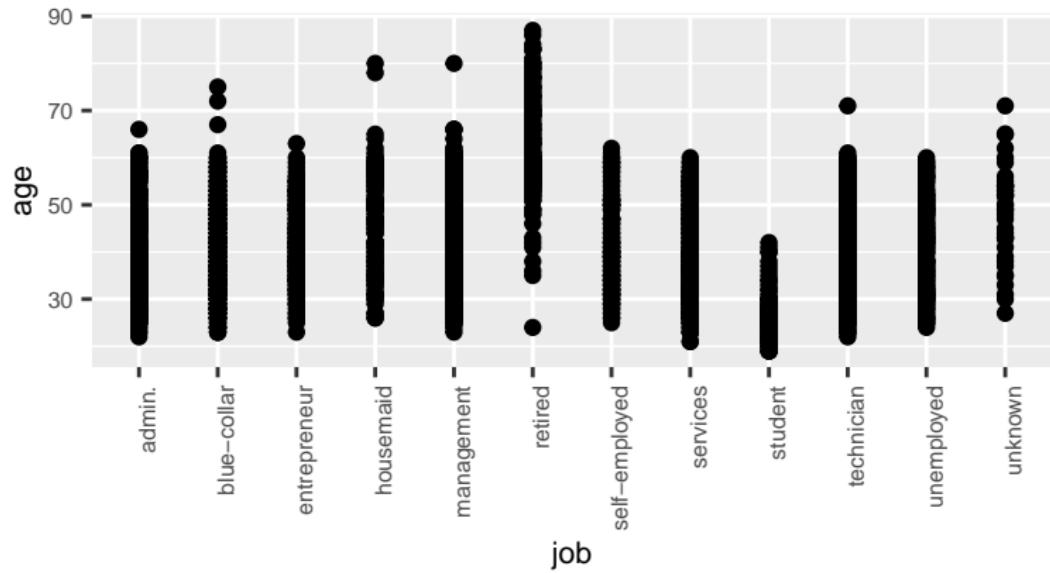
For non-numeric data: age and default (make it landscape)

```
ggplot(bank, aes(age, default)) + geom_point()
```



For non-numeric data: job and age

```
ggplot(bank, aes(job, age)) + geom_point() +  
  theme(text = element_text(size=8),  
        axis.text.x = element_text(angle=90, hjust=1))
```



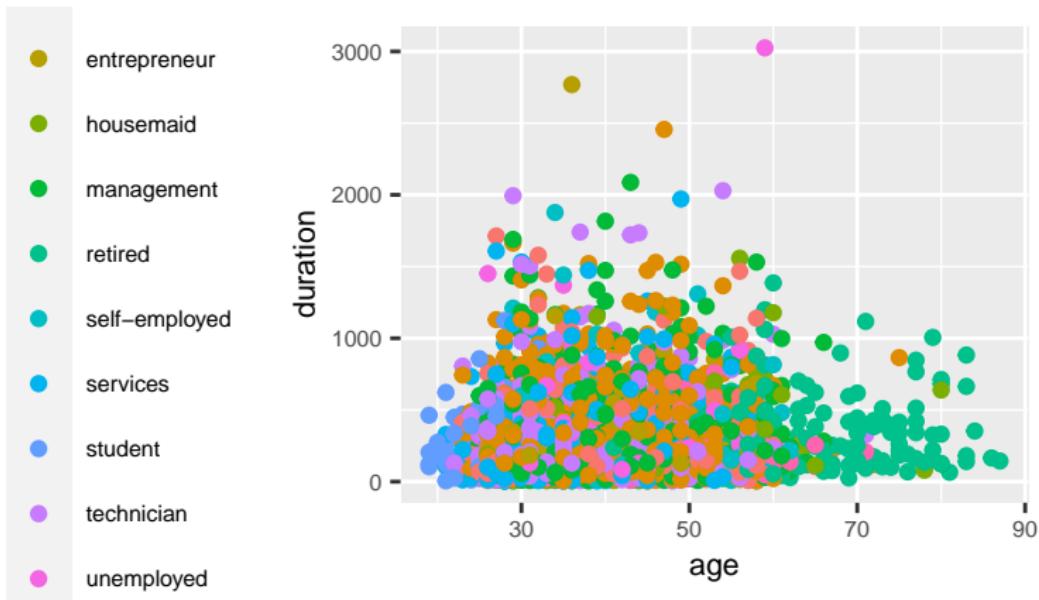
Adjustment: legend bottom

```
# adjust legend position  
ggplot(bank, aes(x = age, y = duration, color = job)) +  
  geom_point() +  
  theme(legend.position="bottom", text = element_text(size=8))
```



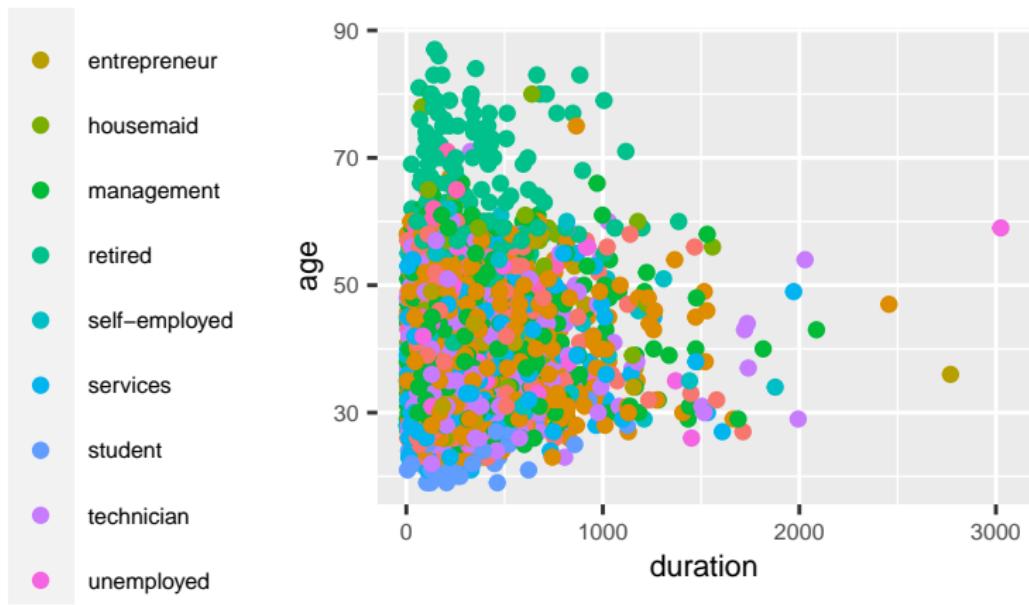
Adjustment: legend left

```
# legend to the left  
ggplot(bank, aes(x = age, y = duration, color = job)) +  
  geom_point() +  
  theme(legend.position="left", text = element_text(size=8))
```



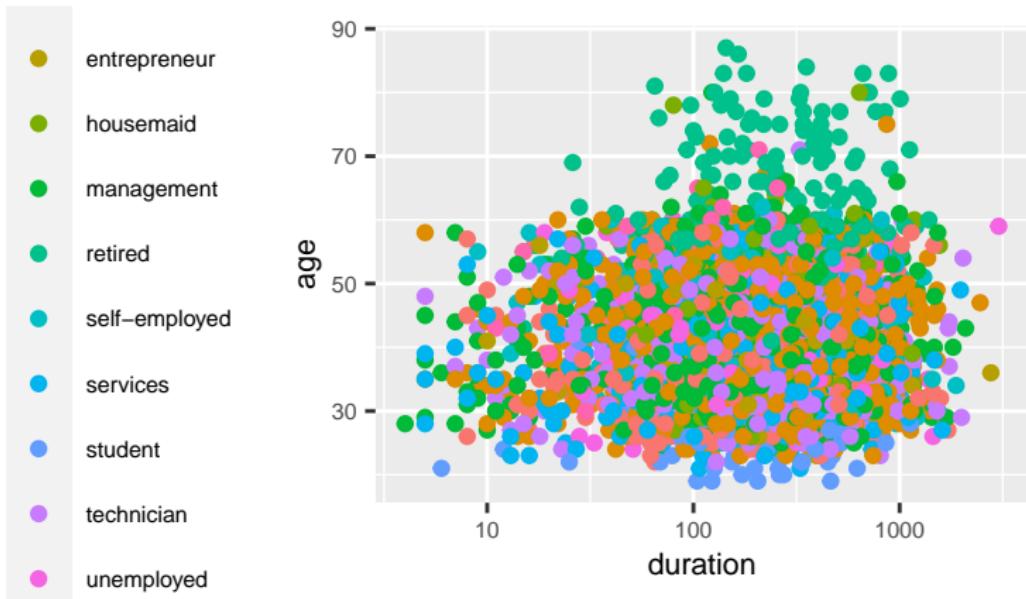
Adjustment: coordinate flip

```
# Flip the x and y axis  
# Different feeling?  
ggplot(bank, aes(x = age, y = duration, color = job)) +  
  geom_point() +  
  theme(legend.position="left", text = element_text(size=8)) +  
  coord_flip()
```



Adjustment: log scale

```
# Make y as log scaled.  
# Note that before flip, x is y, so we use scale_y_log10()  
ggplot(bank, aes(x = age, y = duration, color = job)) +  
  geom_point() +  
  theme(legend.position="left", text = element_text(size=8)) +  
  coord_flip() +  
  scale_y_log10()
```

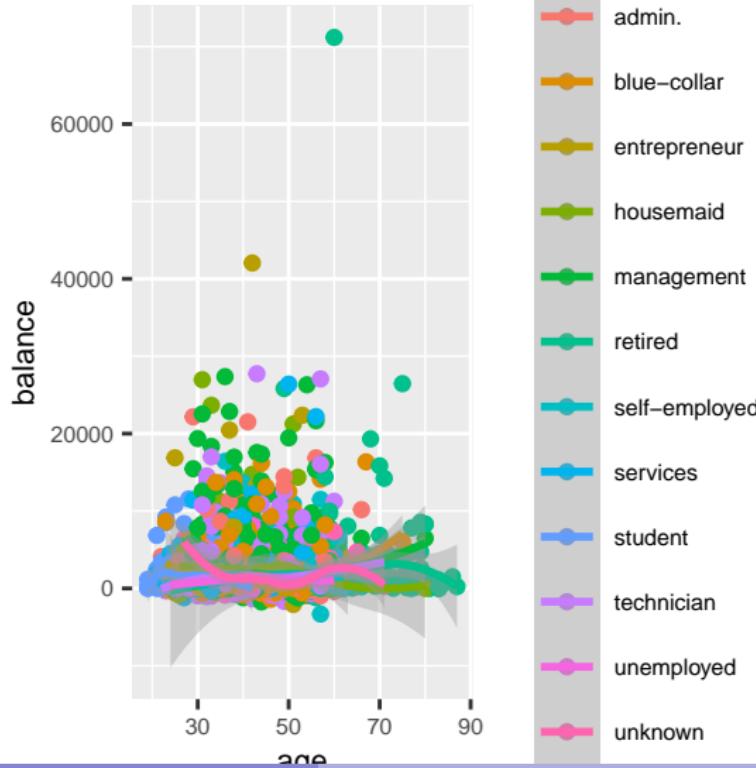
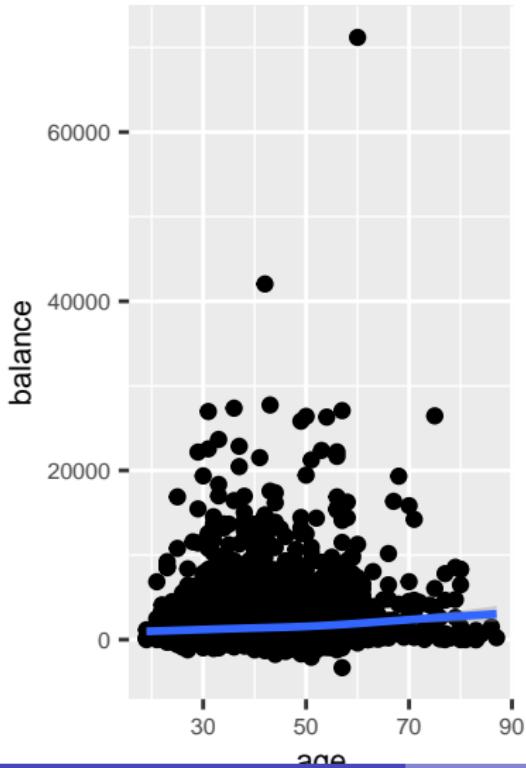


Add 2nd geometry

```
ggplot(bank, aes(age, balance)) + geom_point() + geom_smooth()  
ggplot(bank, aes(age, balance, color = job)) + geom_point() + geom_smooth()
```

Add 2nd geometry - Plot

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'  
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Multiple geoms: Pass aes down

`geom_*` functions has a default parameter of `inherit.aes = TRUE`.

```
ggplot(bank, aes(x = age, y = duration)) +
  geom_smooth() # same as geom_smooth(aes(x = age, y = duration))
  geom_point() # same as geom_point(aes(x = age, y = duration))
```

This is equivalent to below. But this is a bit repeating.

```
ggplot(bank) +
  geom_point(aes(x = age, y = duration)) +
  geom_smooth(aes(x = age, y = duration))
```

Pass aes down or not: Case of Not

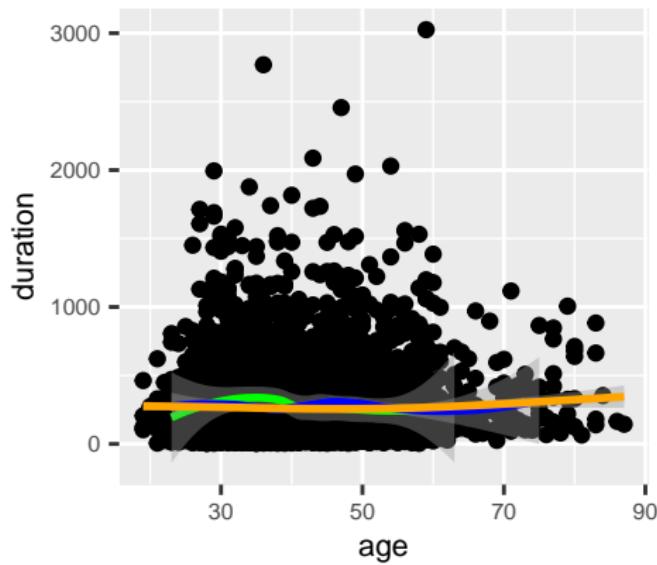
Some cases, we do need to have different data and aes on layers for different geom_* functions.

```
ggplot(bank) +  
  geom_point(aes(x = age, y = duration)) +  
  geom_smooth(data = filter(bank, job == "entrepreneur"),  
              aes(x = age, y = duration), color = "green") +  
  geom_smooth(data = filter(bank, job == "blue-collar"),  
              aes(x = age, y = duration), color = "blue") +  
  geom_smooth(data = filter(bank, job != "entrepreneur"),  
              aes(x = age, y = duration), color = "orange") +
```

Pass aes down or not: Case of Not - Result

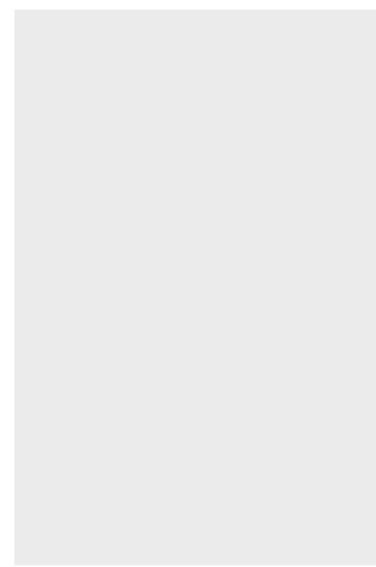
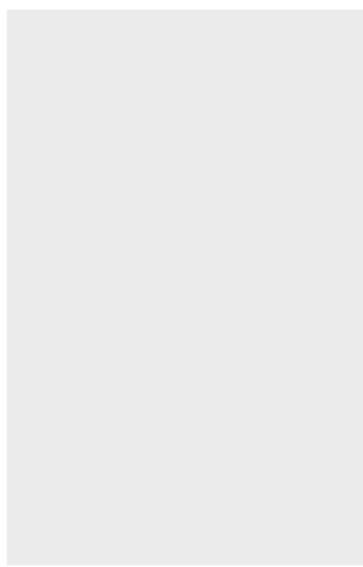
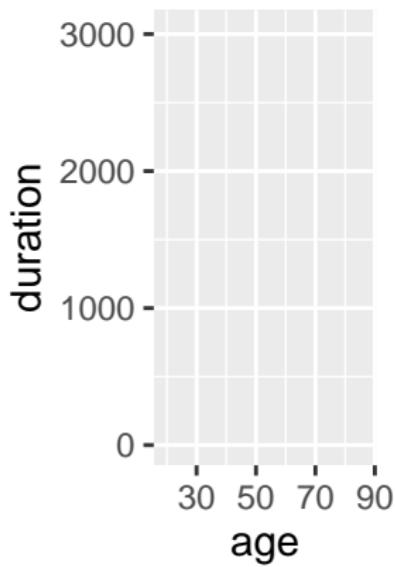
I added `theme(text = element_text(size=8))` to many ggplot statement. This is to fit to the slide size. You may NOT need it in a R Markdown.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'  
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Each + is a layer

```
# Nearly empty chart.  
g <- ggplot(bank, aes(x = age, y = duration))  
# This is almost empty  
g <- ggplot(bank)  
# This is really empty.  
g <- ggplot()
```

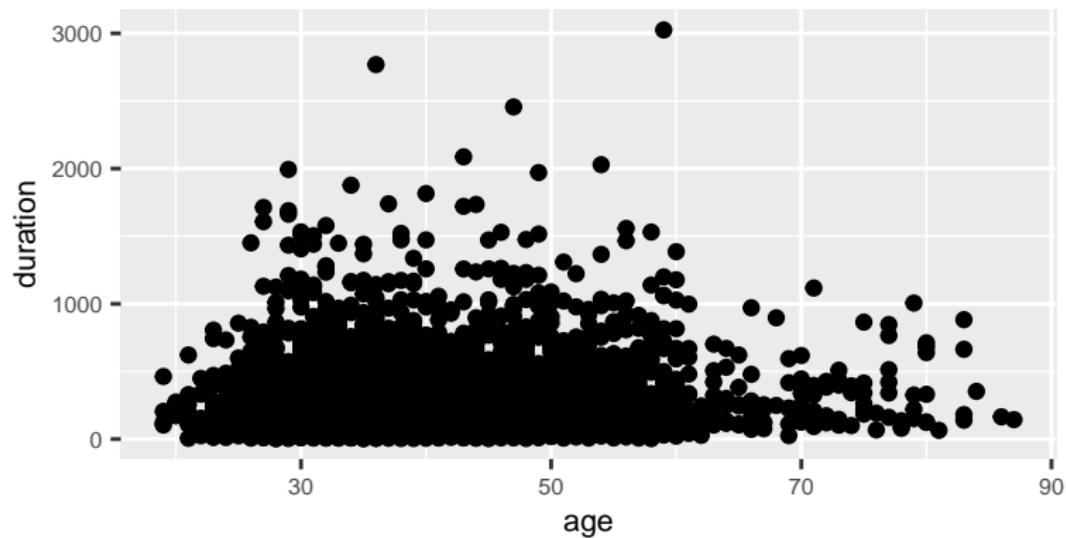


Combine σ with layers

```
ggplot(bank, aes(x = age, y = duration)) + theme(text = element_text(size=8)) +  
  geom_point() + geom_smooth()
```

This is equivalent to above

```
g <- ggplot(bank, aes(x = age, y = duration)) + theme(text = element_text(size=8))  
g + geom_point()
```



```
g + geom_line()
```

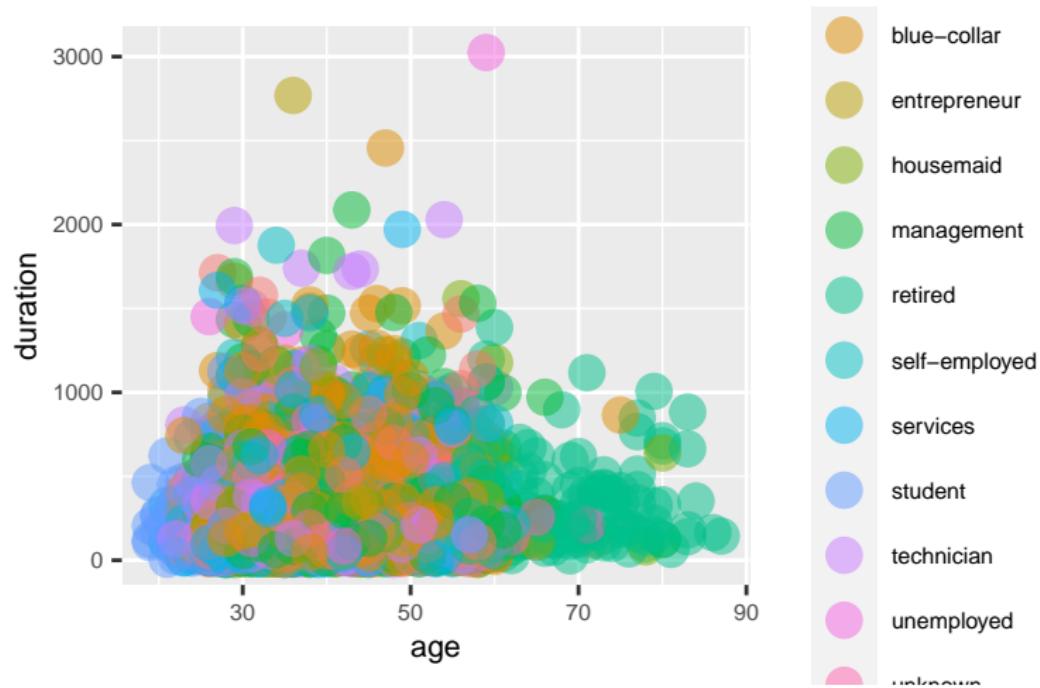
g can be re-used

- g can be re-used. It's good to be used when we want to explore data and try to plot many figures.
- Fixed a few variables in `g <- ggplot(data, aes(job, balance))`.
- Use `g + geom_XXX()` to find the best representation for the relationship between variables.

```
g <- ggplot(bank, aes(job, balance))
g + geom_point()
g + ()
```

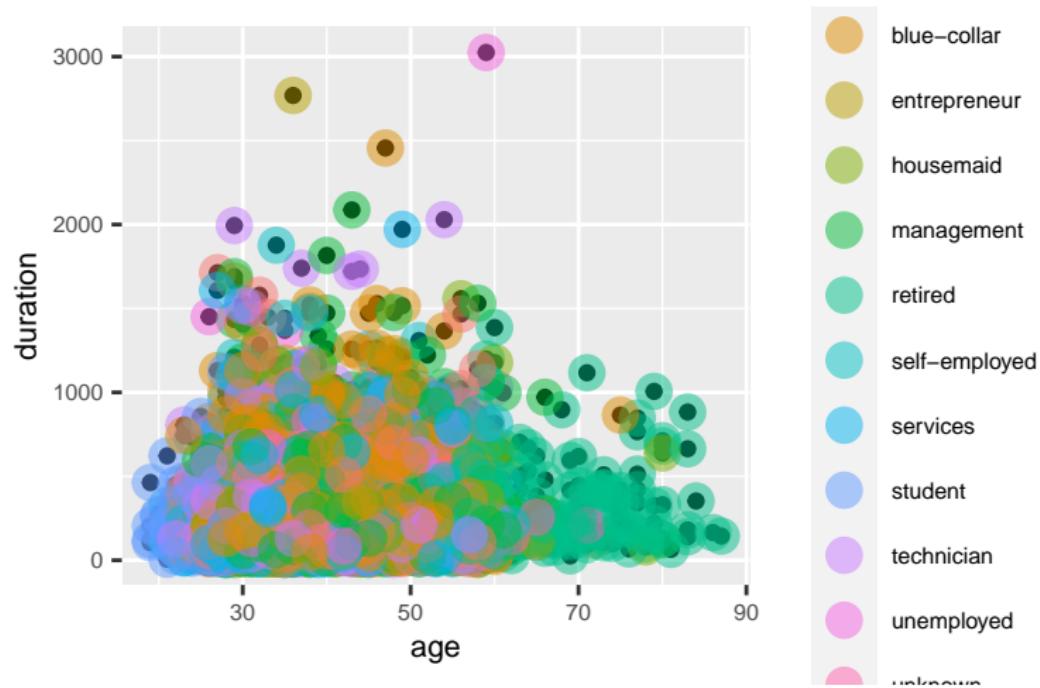
g: mix and match: which figure?

1. `g + geom_point() + geom_smooth(method = "lm") + facet_grid(. ~ job)`
2. `g + geom_point(color = "steelblue", size = 4, alpha = 1/2)`
3. `g + geom_point(aes(color = job), size = 4, alpha = 1/2)`
4. `g + geom_point() + geom_point(aes(color = job), size = 4, alpha = 1/2)`



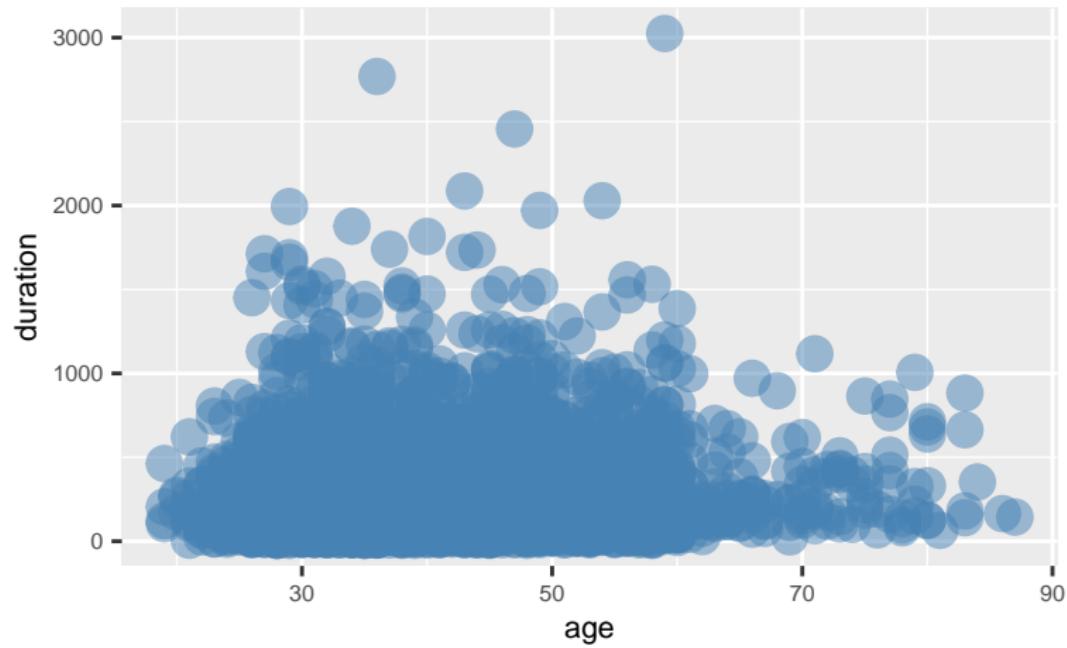
g: mix and match: which figure?

1. `g + geom_point() + geom_smooth(method = "lm") + facet_grid(. ~ job)`
2. `g + geom_point(color = "steelblue", size = 4, alpha = 1/2)`
3. `g + geom_point(aes(color = job), size = 4, alpha = 1/2)`
4. `g + geom_point() + geom_point(aes(color = job), size = 4, alpha = 1/2)`



g: mix and match: which figure?

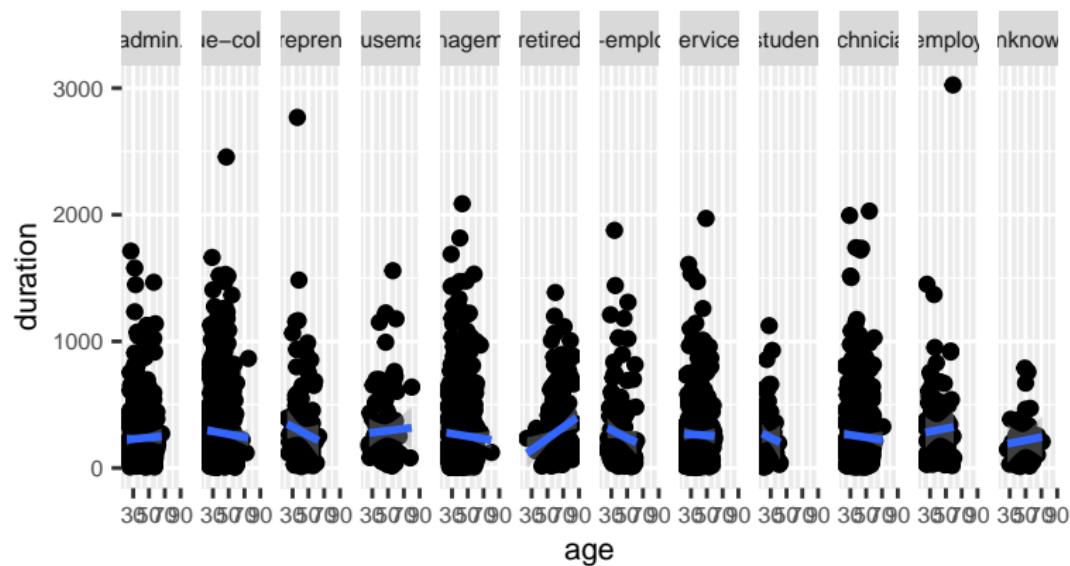
1. `g + geom_point() + geom_smooth(method = "lm") + facet_grid(. ~ job)`
2. `g + geom_point(color = "steelblue", size = 4, alpha = 1/2)`
3. `g + geom_point(aes(color = job), size = 4, alpha = 1/2)`
4. `g + geom_point() + geom_point(aes(color = job), size = 4, alpha = 1/2)`



g: mix and match: which figure?

1. `g + geom_point() + geom_smooth(method = "lm") + facet_grid(. ~ job)`
2. `g + geom_point(color = "steelblue", size = 4, alpha = 1/2)`
3. `g + geom_point(aes(color = job), size = 4, alpha = 1/2)`
4. `g + geom_point() + geom_point(aes(color = job), size = 4, alpha = 1/2)`

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



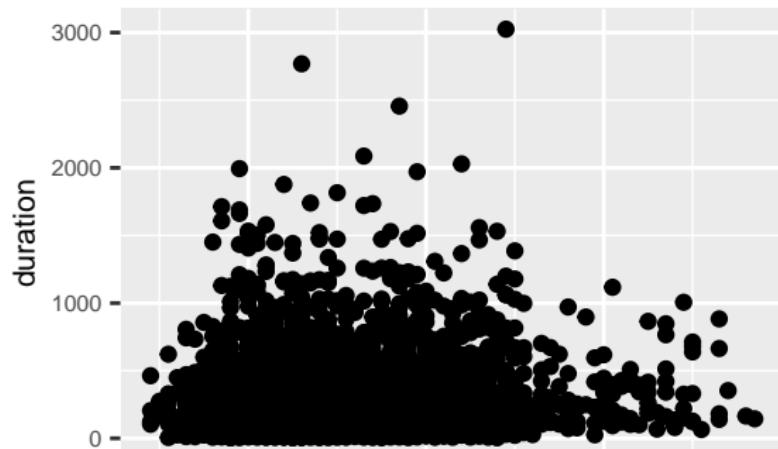
aes and aes_string

- aes needs to have fixed column names. use aes_string to pass variable name as string.

```
ggplot(bank, aes_string("age", "balance", color = "job")) + geom_point()
```

Example of aes_string

```
# Instead of, ggplot(bank, aes(age, balance)) + geom_point()
plot_bank <- function(x, y) {
  ggplot(bank, aes_string(x, y)) + geom_point() + theme(text = element_text(size=8))
}
plot_bank("age", "duration")
```



Example for aes_string

You may use `aes_string` in Shiny app to get user's input.

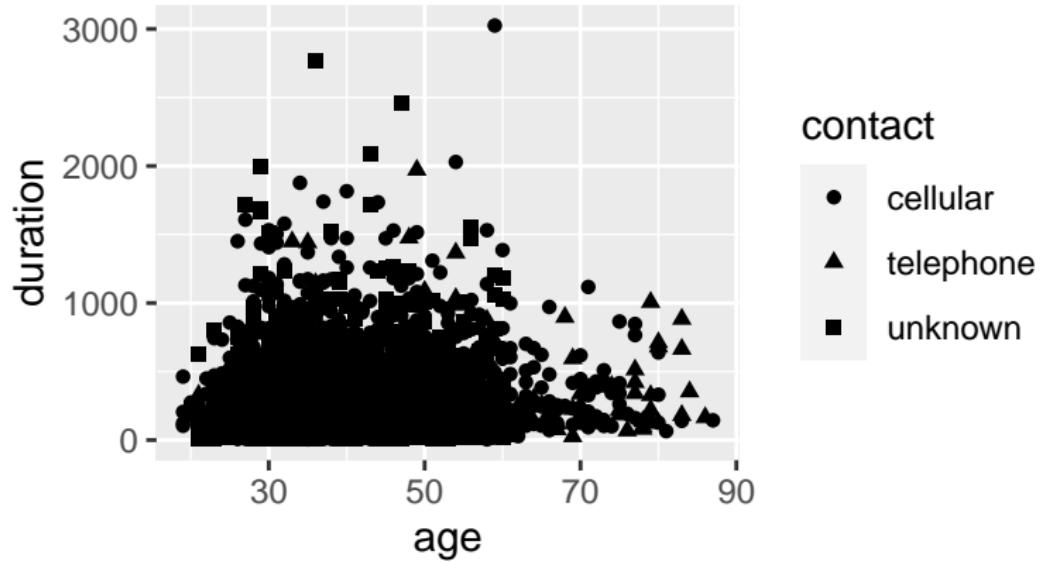
```
# in a Shiny app
# in ui
selectInput("column", "col", choices = c("balance", "duration"))
# in server
plot_bank("age", input$col)
```

Differentiate groups

- shape
- color
- size
- alpha

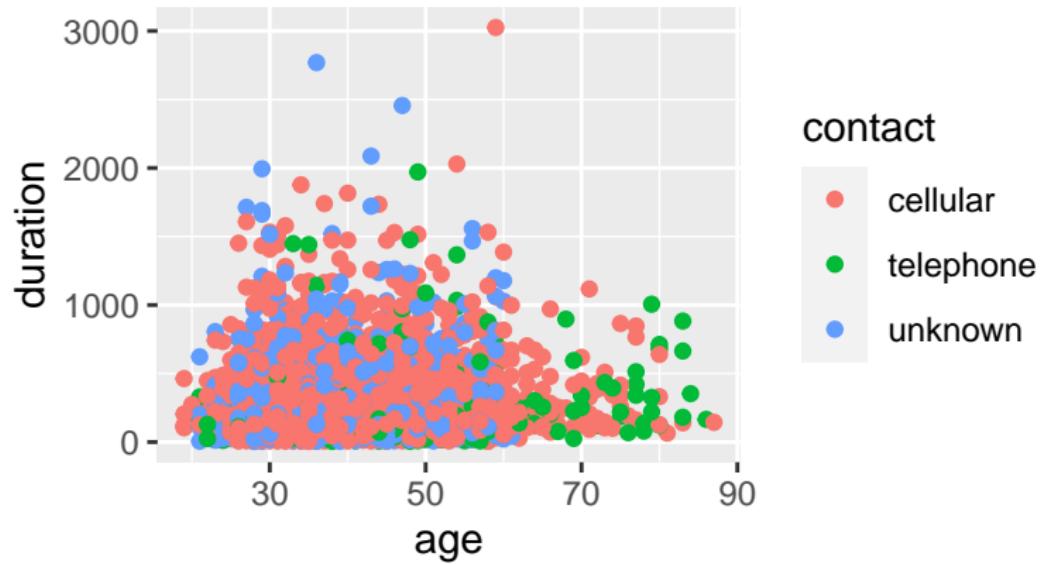
Differentiate groups - shape

```
# Use shape  
ggplot(bank) +  
  geom_point(aes(age, duration, shape = contact))
```



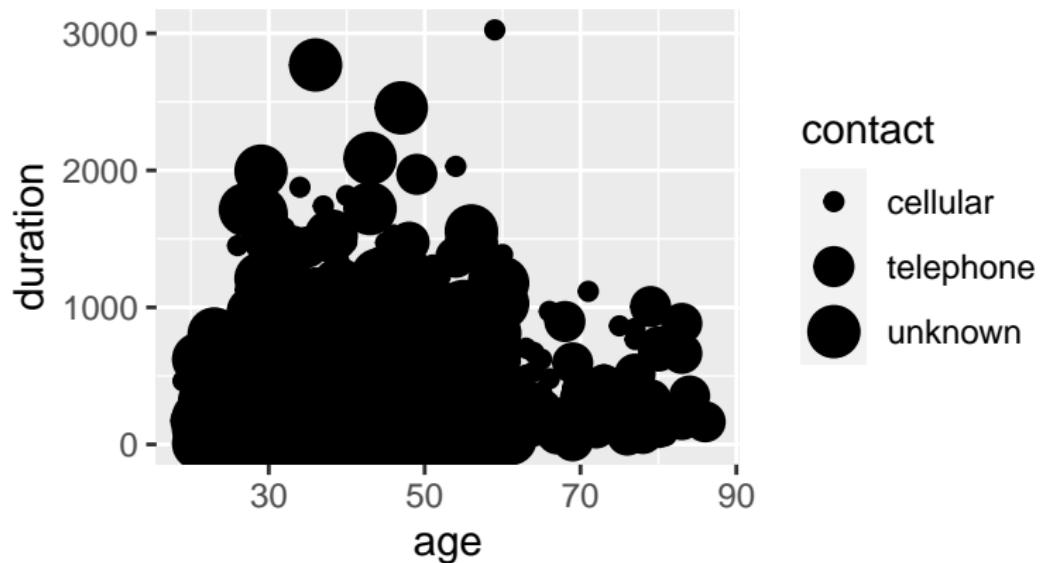
Differentiate groups - color

```
# Use color  
ggplot(bank) +  
  geom_point(aes(age, duration, color = contact))
```



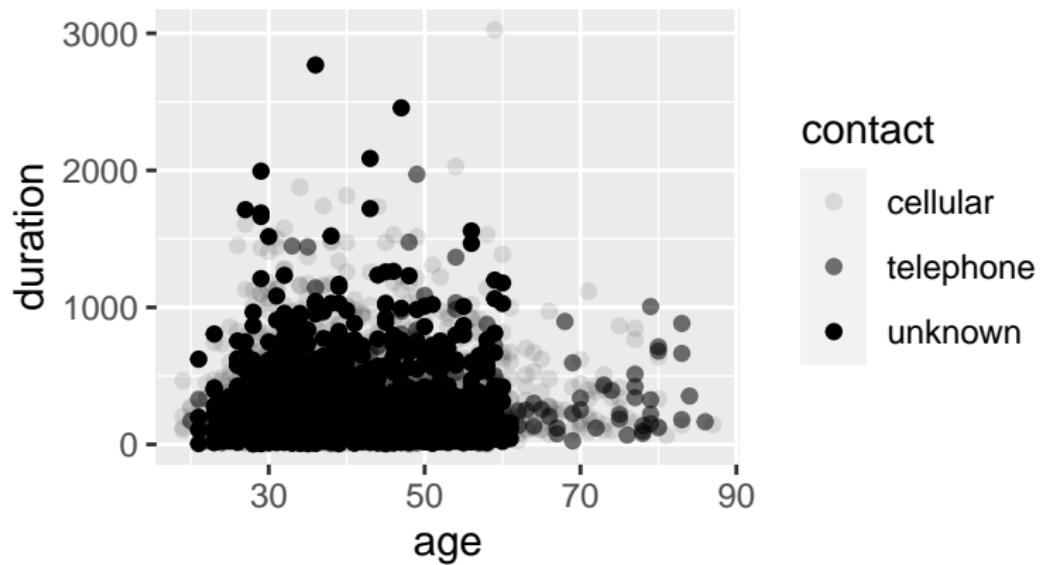
Differentiate groups - size

```
# Use size  
ggplot(bank) +  
  geom_point(aes(age, duration, size = contact))  
## Warning: Using size for a discrete variable is not advised.
```



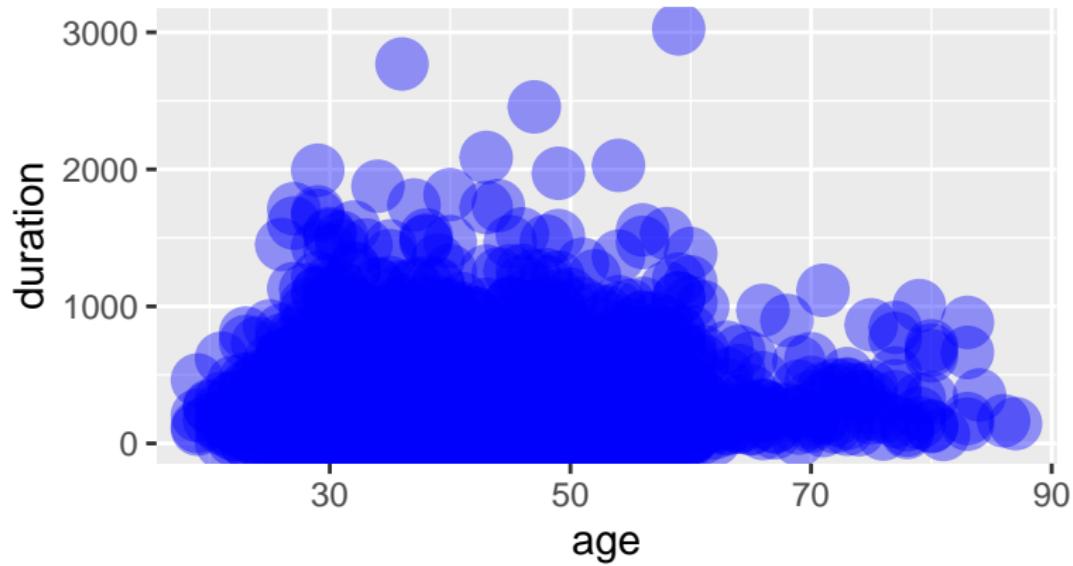
Differentiate groups - alpha

```
# Use alpha - transparency  
ggplot(bank) +  
  geom_point(aes(age, duration, alpha = contact))  
## Warning: Using alpha for a discrete variable is not advised.
```



Enforce color, put things outside aes

```
## you can also enforce color, put things outside aes  
ggplot(bank) +  
  geom_point(aes(age, duration),  
             color = "blue", size = 6, alpha = 0.4)
```



What's inside bank's clients? Things to consider - 1

- Which variables in data are categorical?
- Which variables are continuous?
- Bio:
 - ▶ age
 - ▶ job
 - ▶ marital
 - ▶ education
- Financial
 - ▶ default
 - ▶ balance
 - ▶ housing
 - ▶ loan

What's inside bank's clients? Things to consider - 2

- Communication

- ▶ contact: cellular v.s. telephone v.s. unknown
- ▶ day/month: maybe good to ignore?
- ▶ duration:
- ▶ campaign:
- ▶ pdays:
- ▶ previous:
- ▶ poutcome:

Categorical/Continuous variable

- numeric -> continuous data
- string -> discrete data

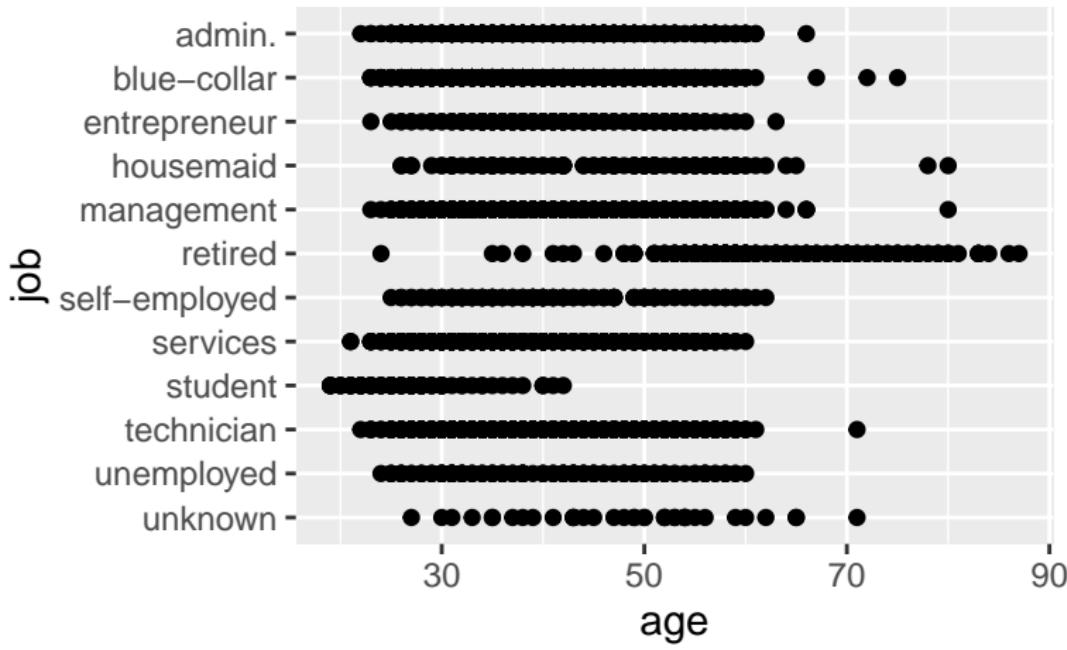
For different type of variables, the scale is also different

- To reverse a categorical (discrete) variable, we use `scale_y_discrete(limit = rev(levels(factor(...))))`.
- To reverse a continuous numerical variable, we use `scale_x_reverse()`.

Exercise

- What does this figure mean?

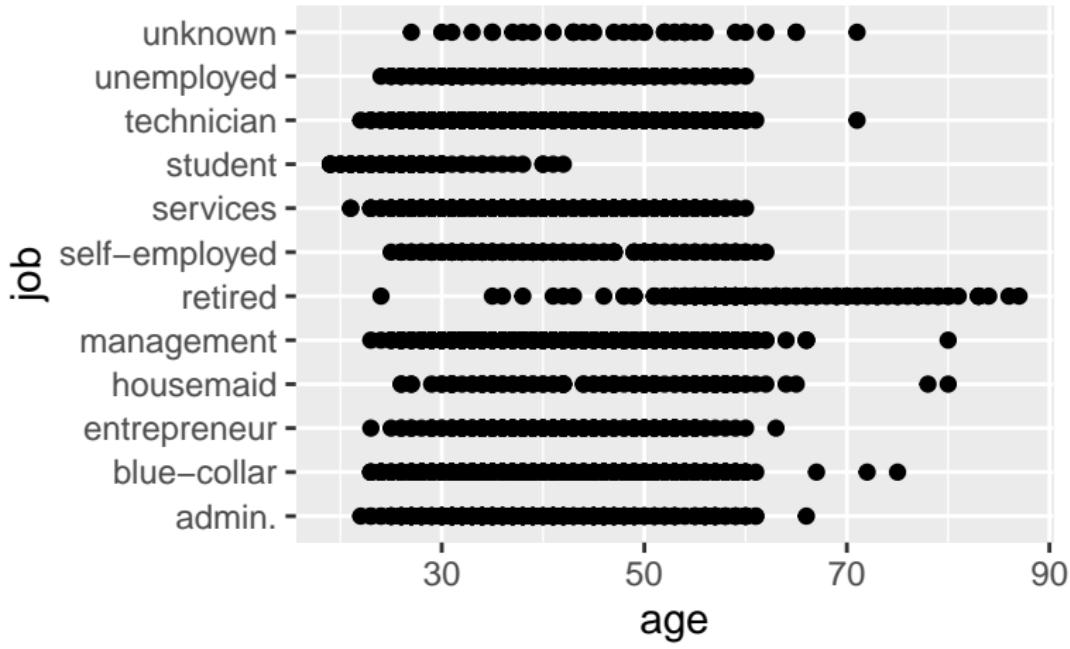
```
ggplot(bank, aes(age, job)) +  
  geom_point() +  
  scale_y_discrete(limit = rev(levels(factor(bank$job))))
```



Exercise

y labels without sort.

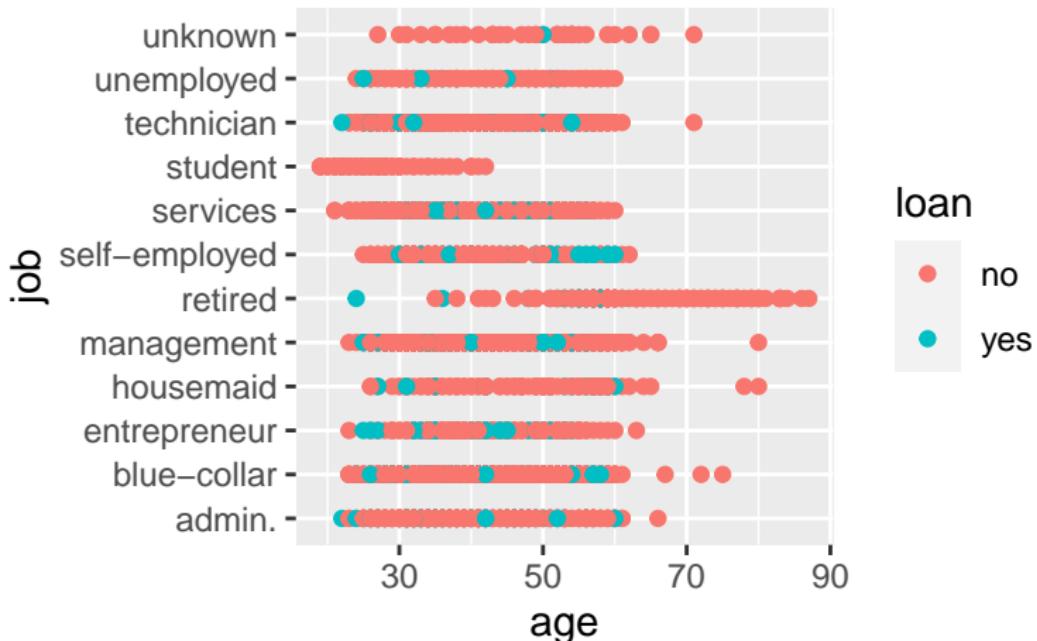
```
ggplot(bank, aes(age, job)) + geom_point()
```



Exercise

- I tried to plot between job, loan and age. Any better idea?

```
suppressWarnings(  
  ggplot(bank, aes(age, job, color = loan)) + geom_point()  
)
```

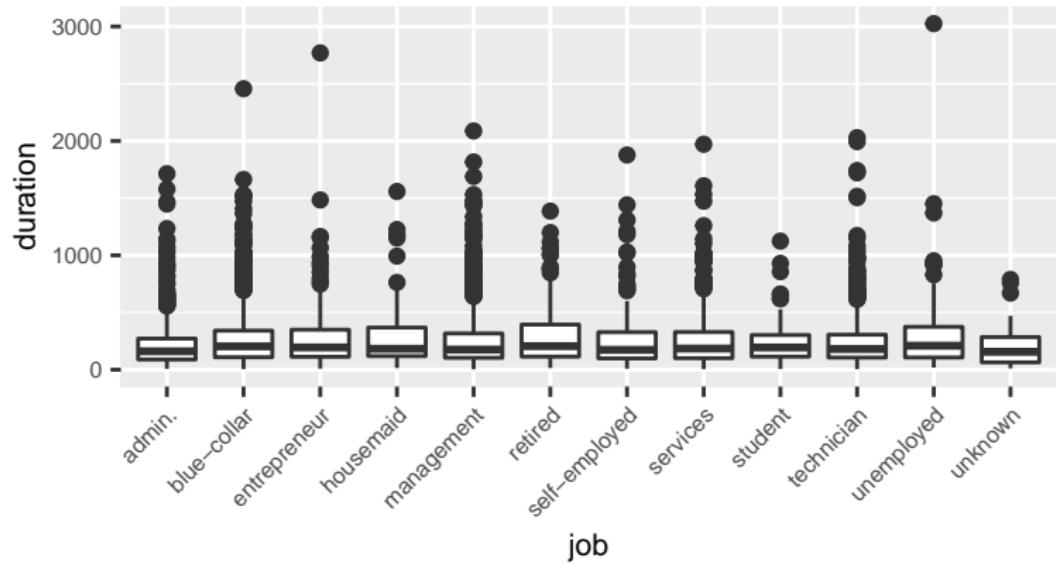


Other geoms

- geom_boxplot()
- geom_density()
- geom_histogram()
- geom_bar()

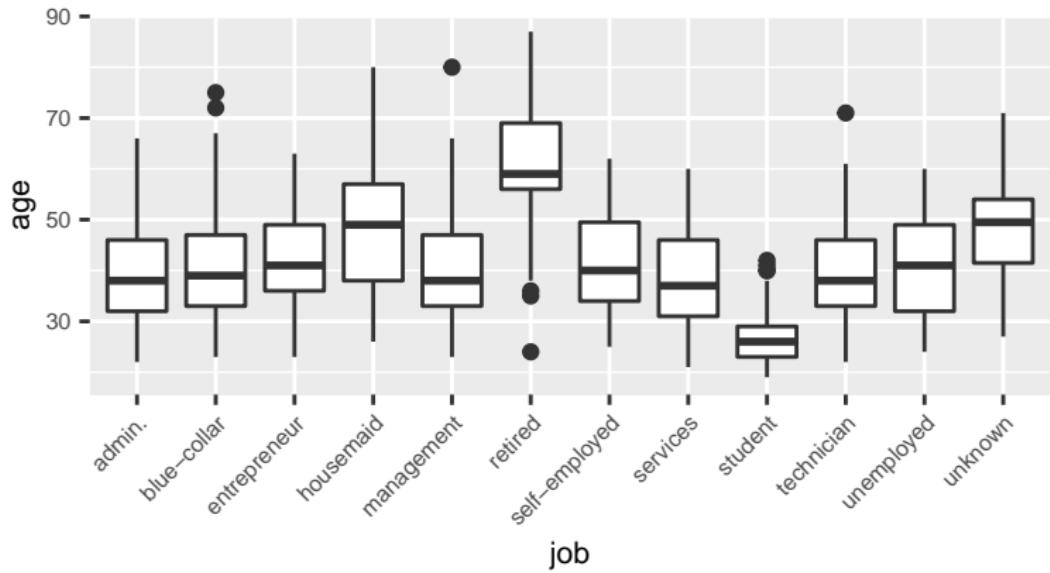
Boxplot: job and duration

```
ggplot(bank, aes(job, duration)) + geom_boxplot() +  
  theme(text = element_text(size=8),  
        axis.text.x = element_text(angle = 45, hjust = 1))
```



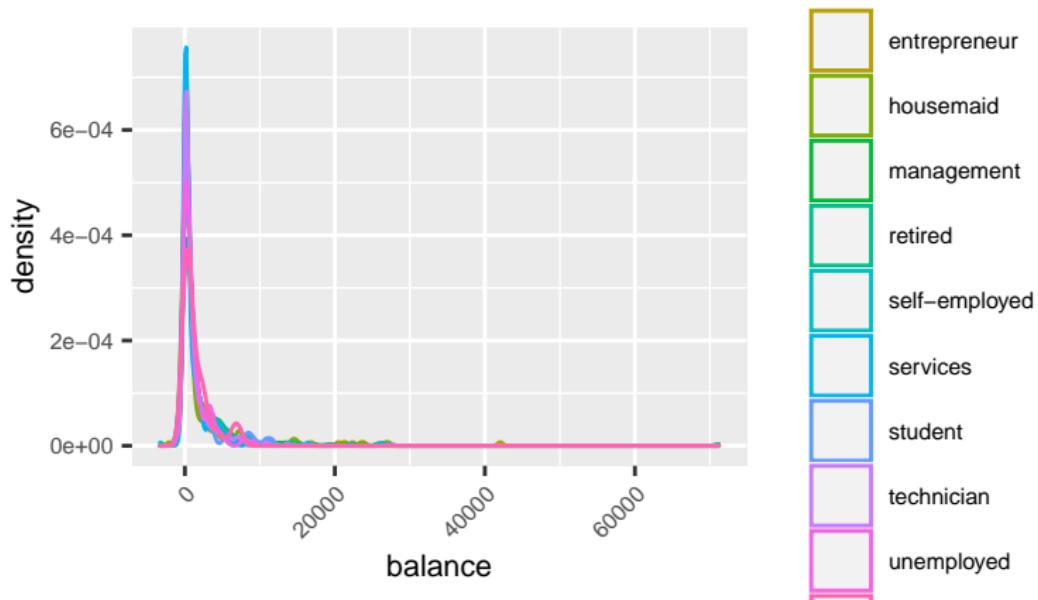
Boxplot: job and age

```
ggplot(bank, aes(job, age)) + geom_boxplot() +  
  theme(text = element_text(size=8),  
        axis.text.x = element_text(angle = 45, hjust = 1))
```



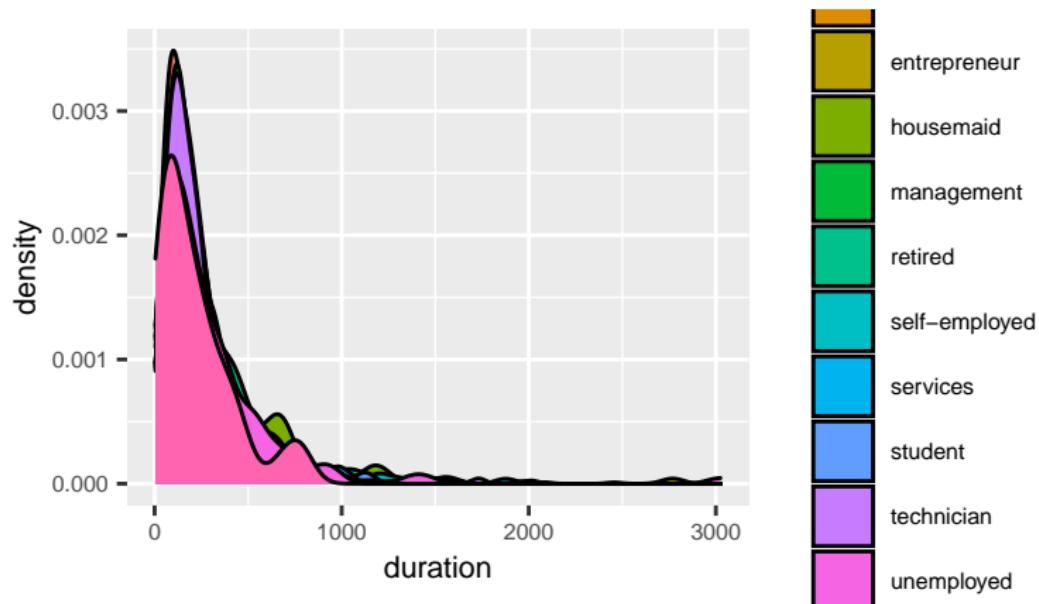
Density: balance and job with color

```
ggplot(bank, aes(balance, color = job)) + geom_density() +  
  theme(text = element_text(size=8),  
        axis.text.x = element_text(angle = 45, hjust = 1))
```



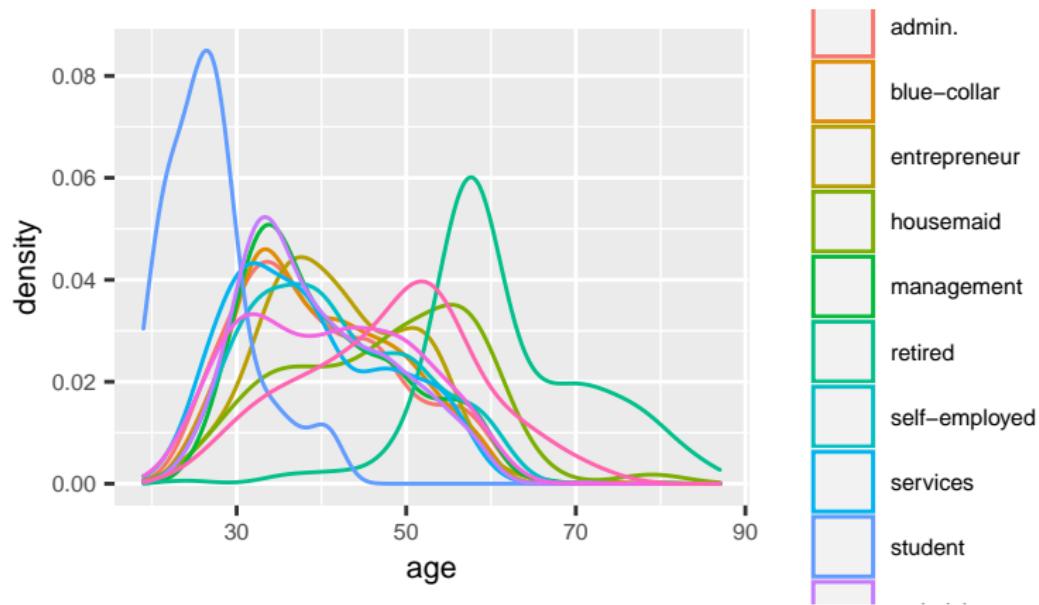
Density: balance and job with fill

```
ggplot(bank, aes(duration, fill = job)) + geom_density() +  
  theme(text = element_text(size=8))
```



Density: age and job with color and alpha

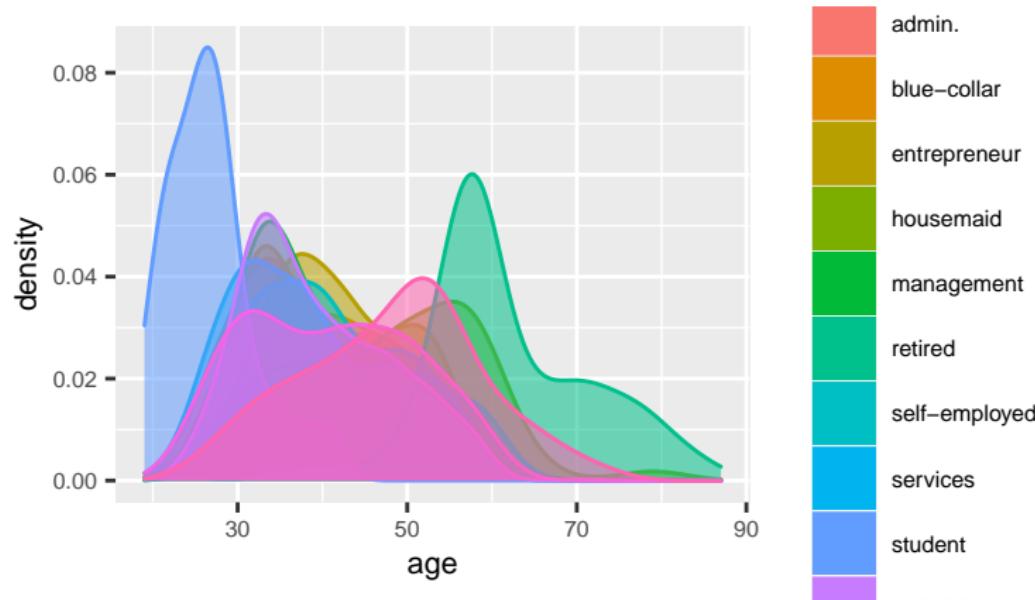
```
ggplot(bank, aes(age, color = job, alpha = 0.3)) + geom_density() +  
  theme(text = element_text(size=8))
```



Density: age and job with fill and color and alpha

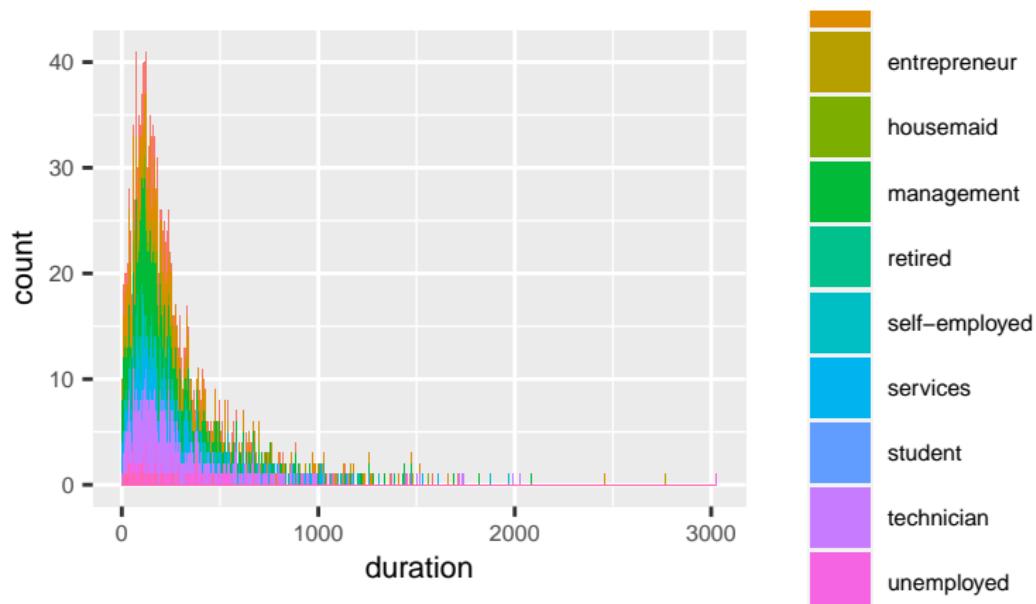
Which is better?

```
ggplot(bank, aes(age, color = job, fill = job, alpha = 0.3)) +  
  geom_density() + theme(text = element_text(size=8))
```



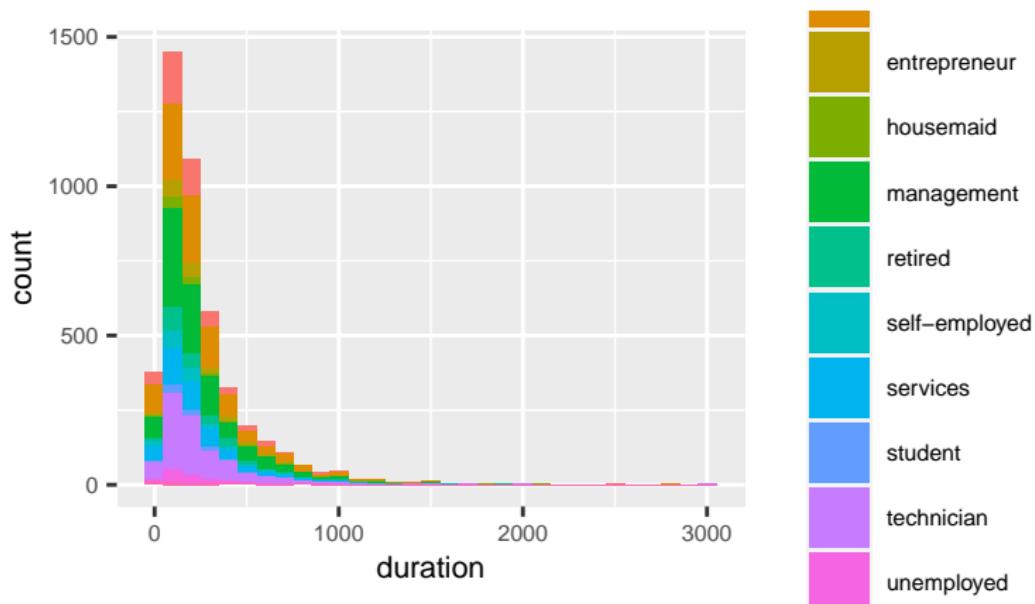
histogram: duration and job and binwidth = 2

```
ggplot(data = bank, mapping = aes(x = duration, fill = job)) +  
  geom_histogram(binwidth = 2) + theme(text = element_text(size=8))
```



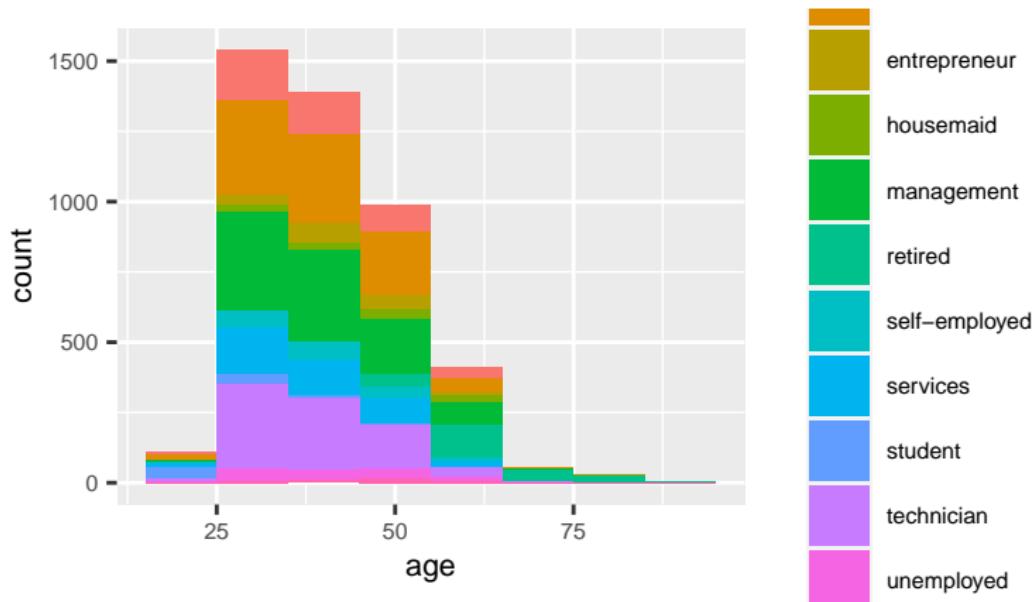
histogram: duration and job and binwidth = 100

```
ggplot(data = bank, mapping = aes(x = duration, fill = job)) +  
  geom_histogram(binwidth = 100) + theme(text = element_text(size=8))
```



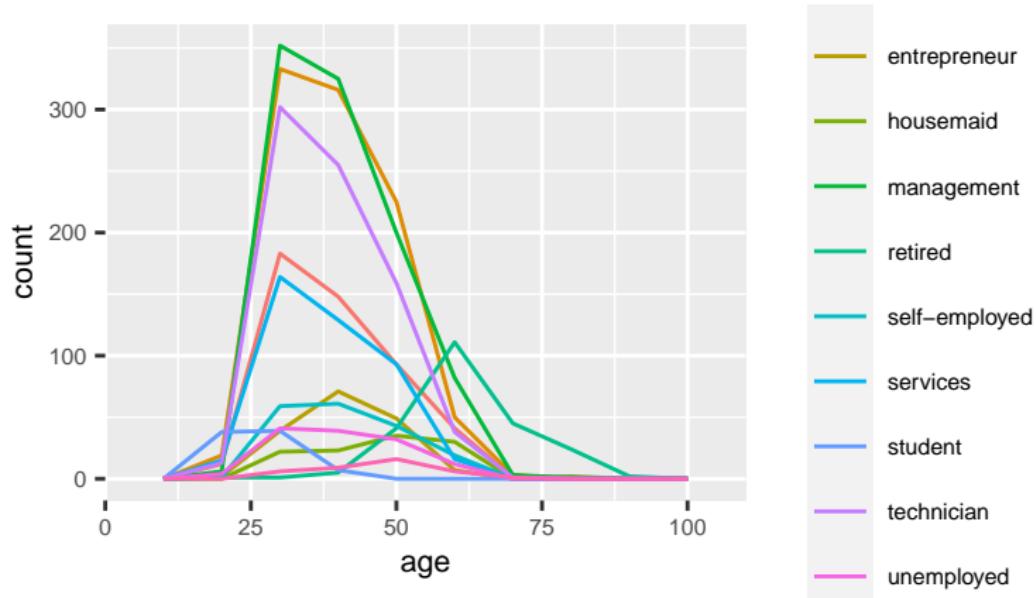
histogram: duration and job and binwidth = 10

```
ggplot(data = bank, mapping = aes(x = age, fill = job)) +  
  geom_histogram(binwidth = 10) + theme(text = element_text(size=8))
```



histogram: age and job and colour

```
ggplot(data = bank, mapping = aes(x = age, colour = job)) +  
  geom_freqpoly(binwidth = 10) + theme(text = element_text(size=8))
```



geom_bar

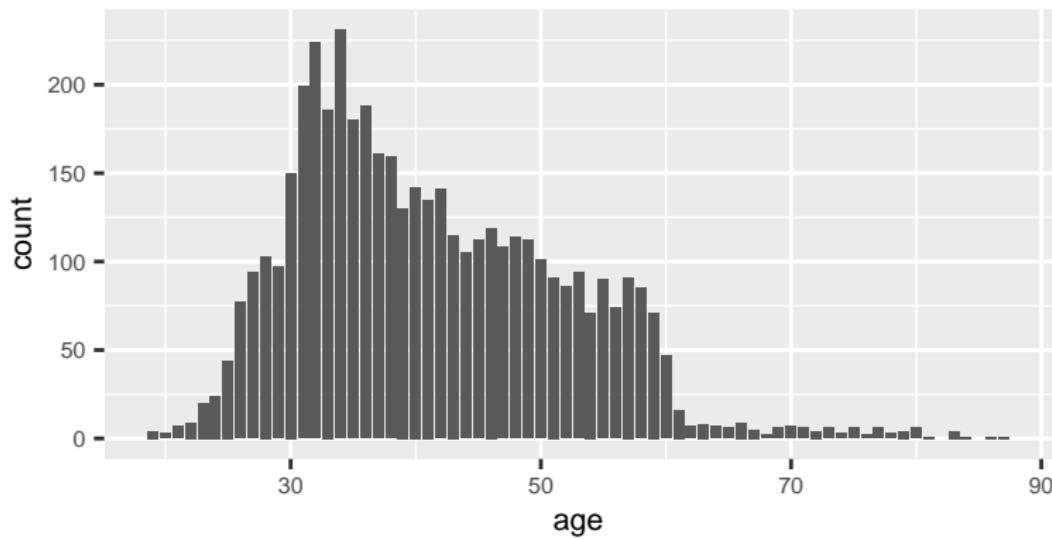
- bar is a statistical function: It counts.

First input parameter to geom_bar is mapping, so we can skip it.

```
ggplot(bank) + geom_bar(mapping = aes(x = age)) +  
  theme(text = element_text(size=8))
```

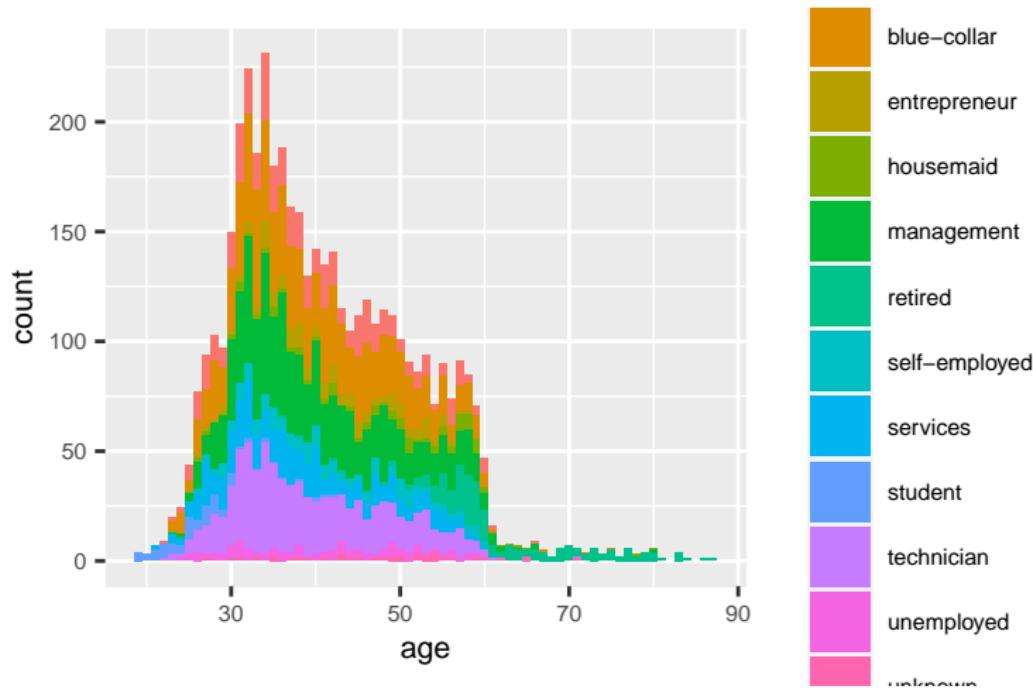
We can skip mapping

```
ggplot(bank) + geom_bar(aes(x = age)) +  
  theme(text = element_text(size=8))
```



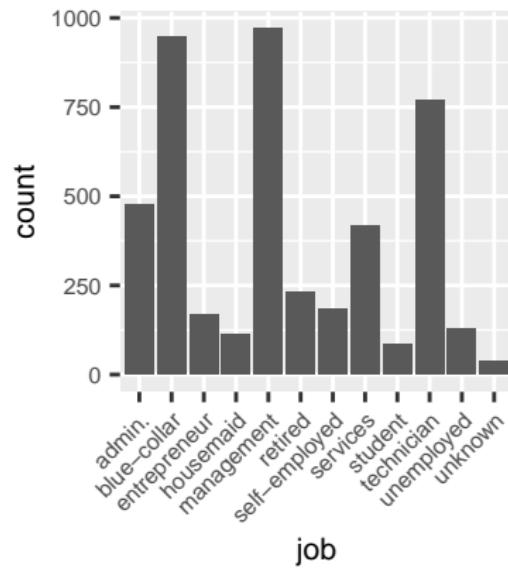
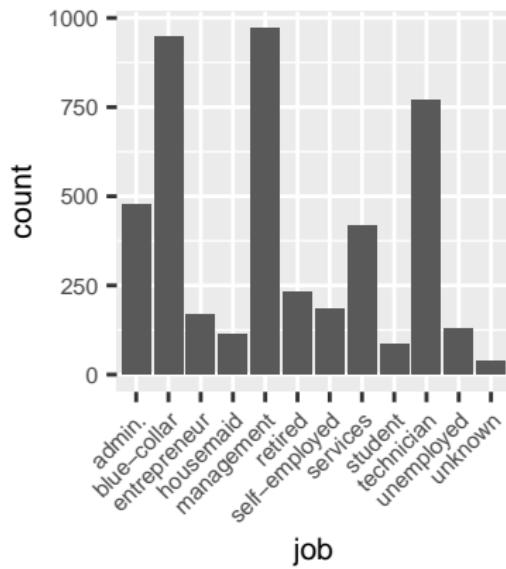
geom_bar with fill: job

```
# comparing to colour, for Bar, we better use fill  
# ggplot(data = bank) + geom_bar(aes(x = age, colour = job))  
ggplot(bank) + geom_bar(mapping = aes(x = age, fill = job)) +  
  theme(text = element_text(size=8))
```



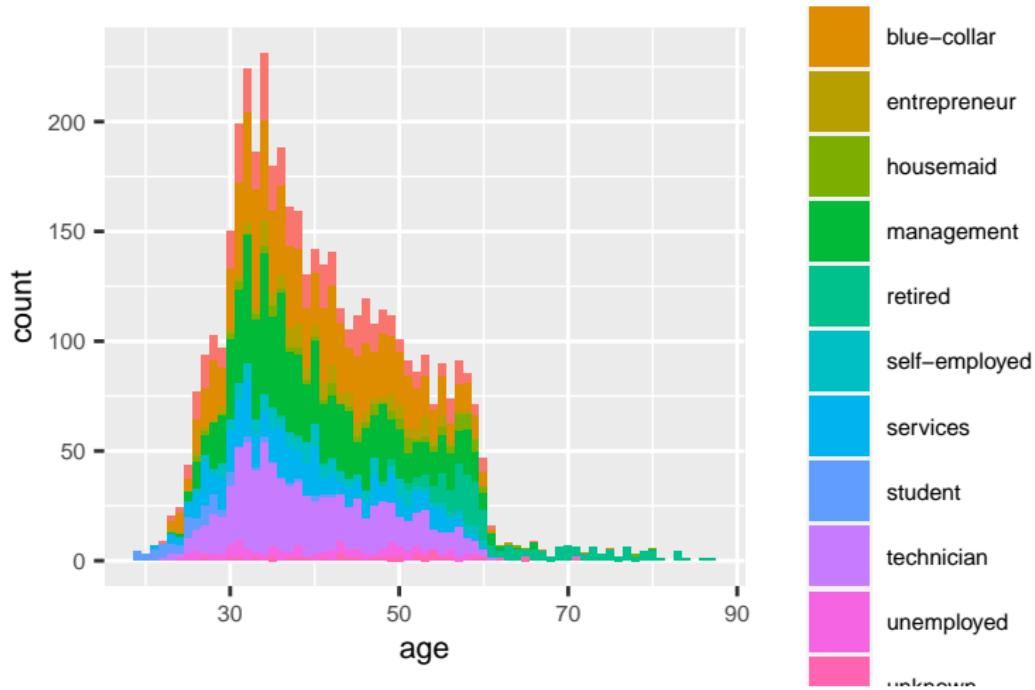
geom_bar with fill: age

```
ggplot(bank) +  
  geom_bar(mapping = aes(x = job)) +  
  theme(text = element_text(size=8))  
# Color doesn't work, because age is a continuous variable.  
ggplot(bank) +  
  geom_bar(mapping = aes(x = job, fill = age)) +  
  theme(text = element_text(size=8))
```



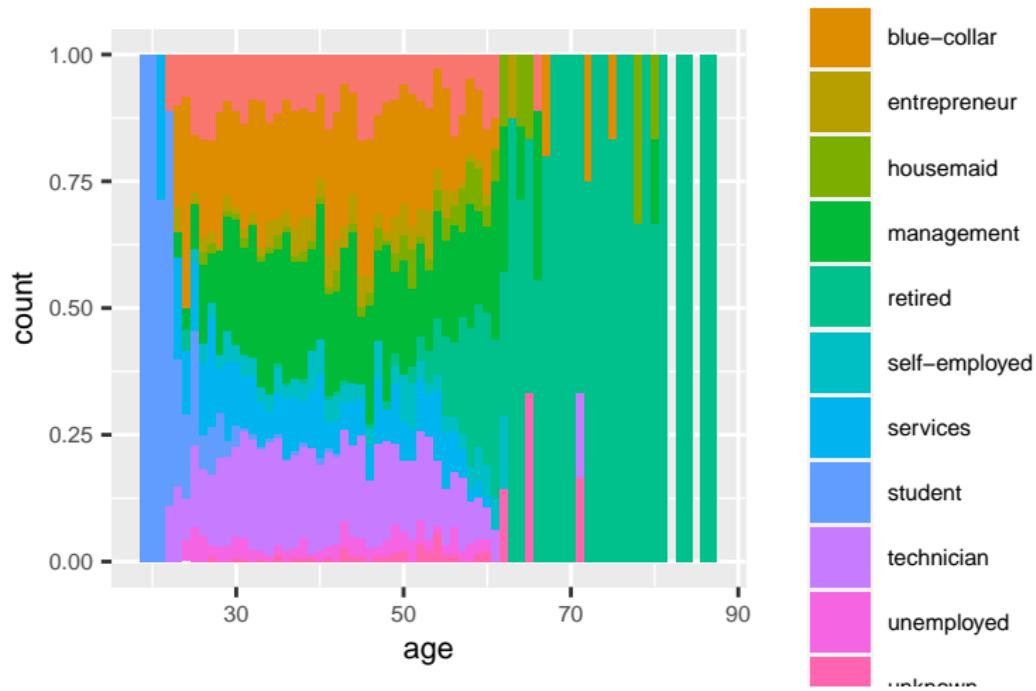
Position for bar

```
ggplot(bank) + geom_bar(mapping = aes(x = age, fill = job)) +  
  theme(text = element_text(size=8))
```



Position for bar: fill

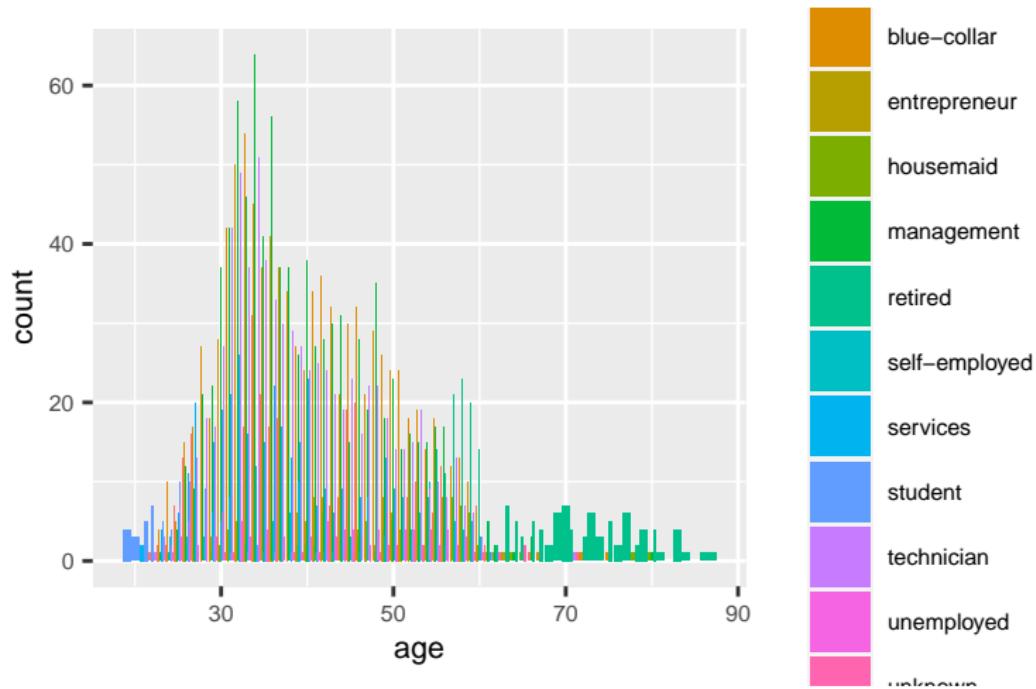
```
# fill to 100%. good to show relative change in different allocations.  
ggplot(bank) + geom_bar(mapping = aes(x = age, fill = job),  
                         position = "fill") +  
  theme(text = element_text(size=8))
```



Position for bar: dodge

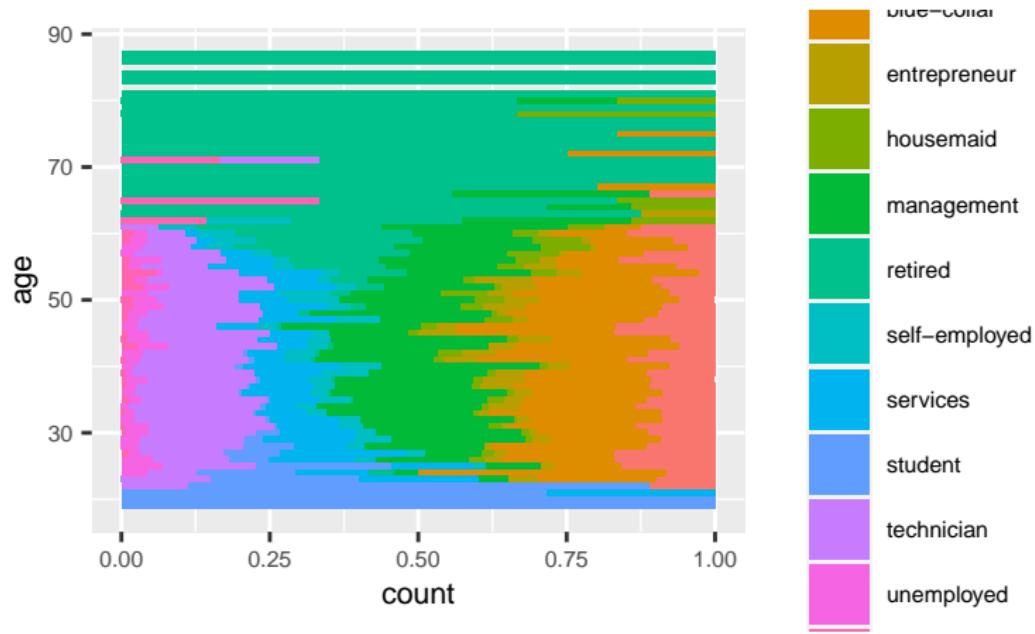
dodge means "adaptive width of the bar".

```
ggplot(bank) + geom_bar(mapping = aes(x = age, fill = job),  
                        position = "dodge") +  
  theme(text = element_text(size=8))
```



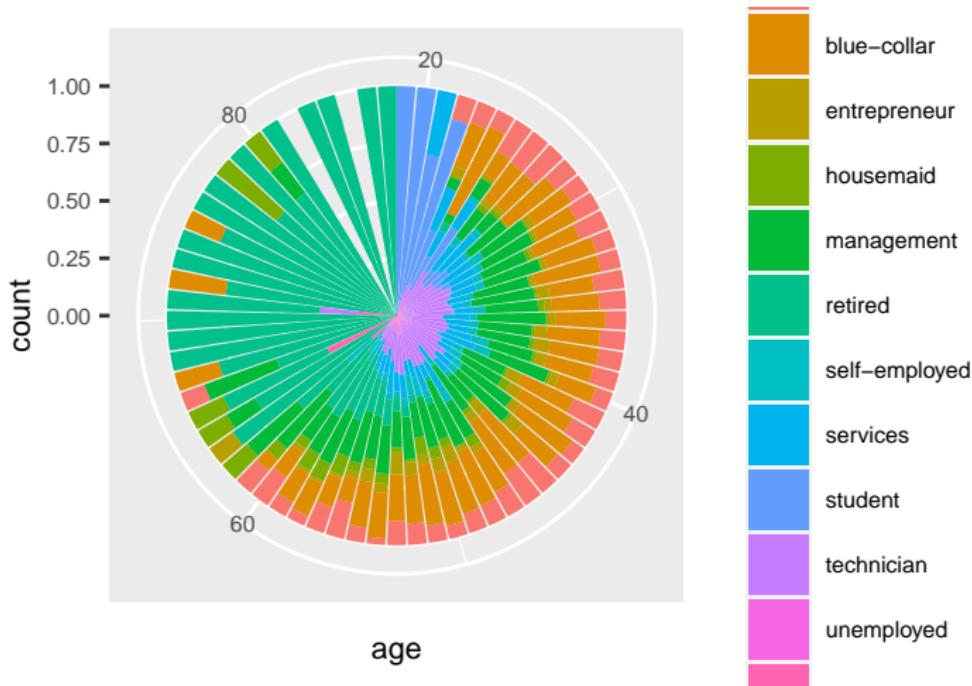
Variations: coord_flip

```
# Switch x and y axis.  
# Note any adjustment on x or y axis is effective on the original name.  
ggplot(bank) +  
  geom_bar(mapping = aes(x = age, fill = job), position = "fill") +  
  coord_flip() + theme(text = element_text(size=8))
```



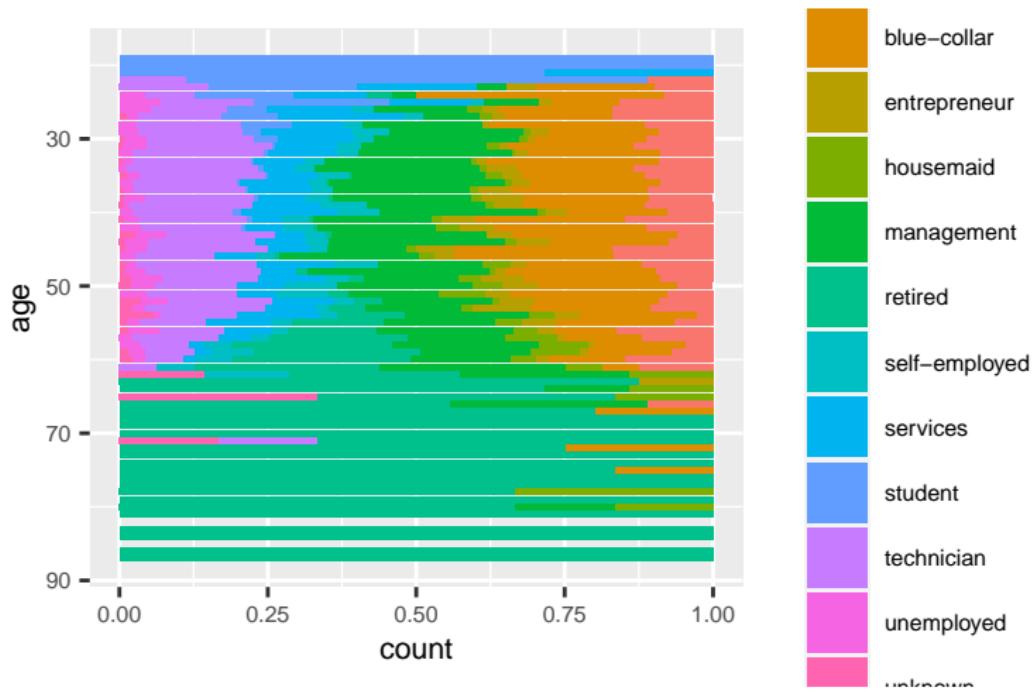
Variations: coord_polar

```
ggplot(bank) +  
  geom_bar(mapping = aes(x = age, fill = job), position = "fill") +  
  coord_polar() + theme(text = element_text(size=8))
```



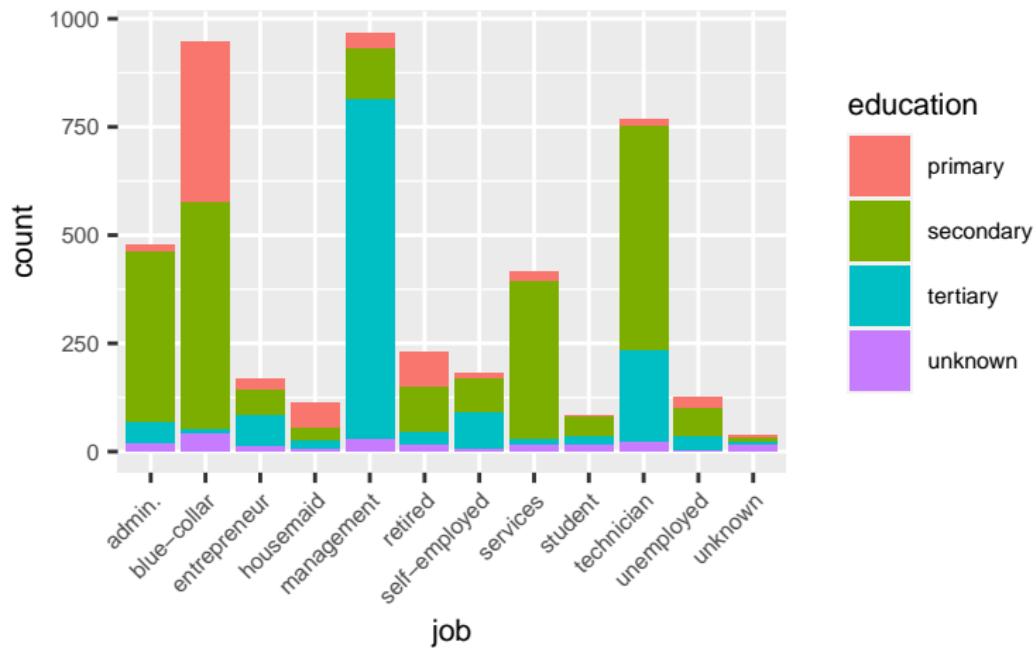
Variations: scale_x_reverse

```
# scale_x_reverse works on continuous variable (numeric, date, etc.). Make it big to .  
ggplot(bank) +  
  geom_bar(mapping = aes(x = age, fill = job), position = "fill") +  
  coord_flip() + scale_x_reverse() + theme(text = element_text(size=8))
```



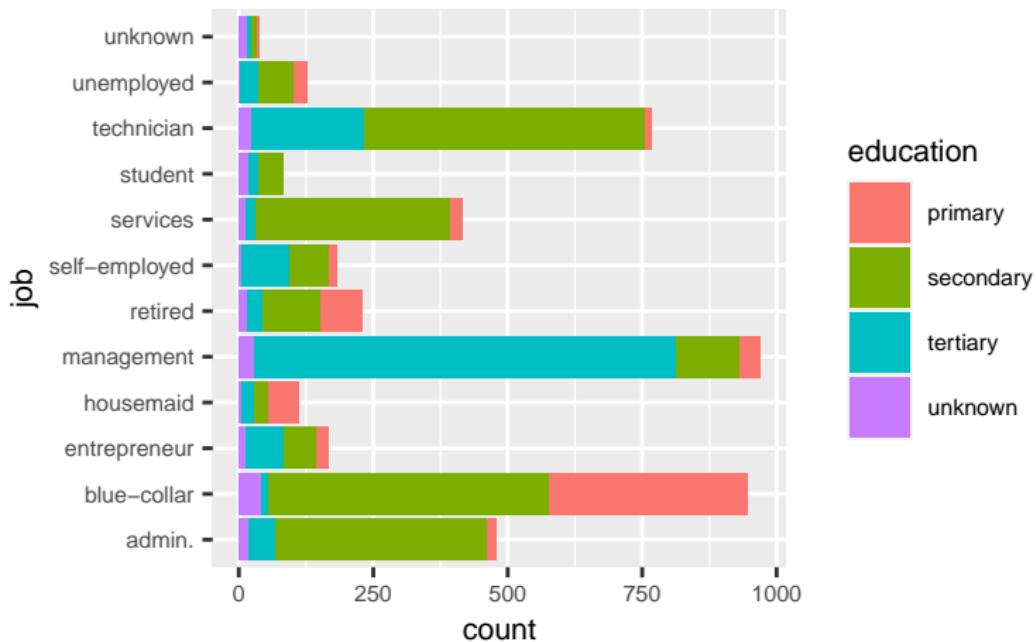
geom_bar: better serves for categorical data

```
ggplot(data = bank, mapping = aes(x = job, fill = education)) +  
  geom_bar() +  
  theme(text = element_text(size=8),  
        axis.text.x = element_text(angle = 45, hjust = 1))
```



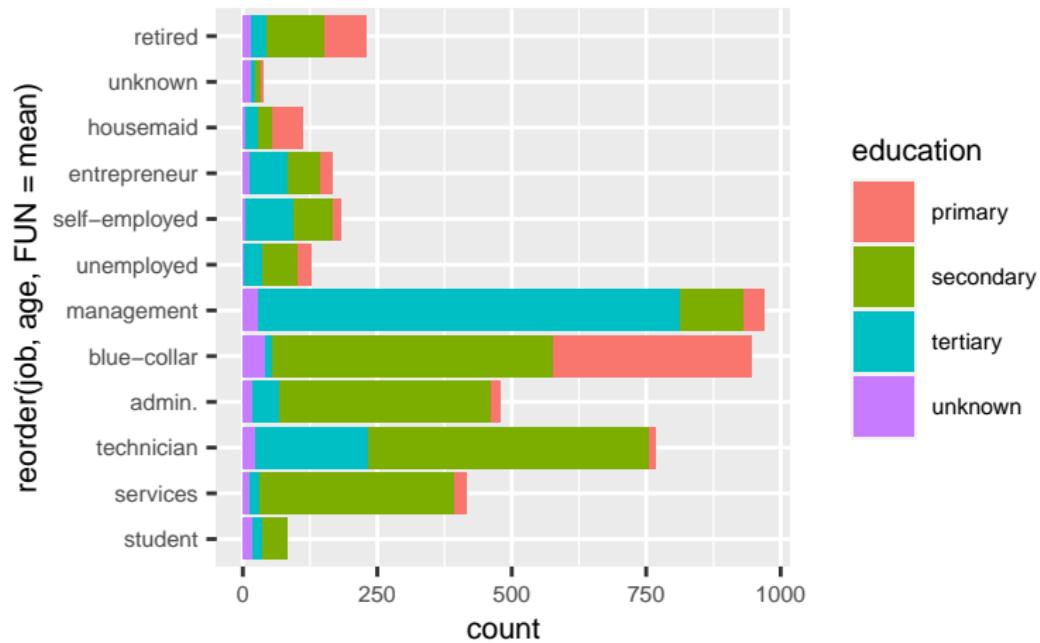
geom_bar: coord_flip

```
ggplot(data = bank, mapping = aes(x = job, fill = education)) +  
  geom_bar() + coord_flip() + theme(text = element_text(size=8))
```



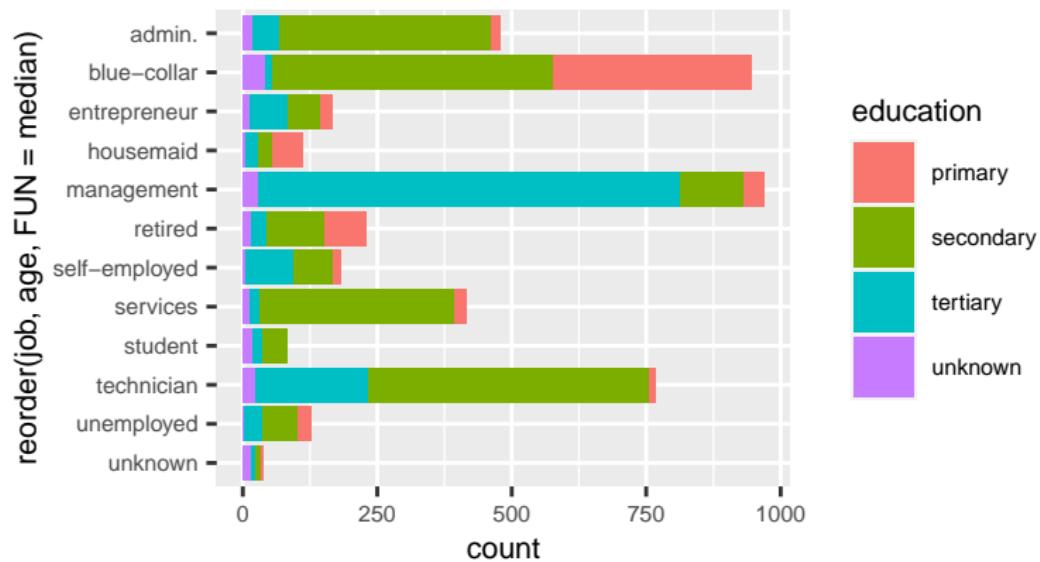
geom_bar: sort job by mean age

```
ggplot(data = bank, mapping = aes(x = reorder(job, age, FUN = mean),  
    fill = education)) + geom_bar() +  
coord_flip() + theme(text = element_text(size=8))
```



geom_bar: sort job by alphabetical order

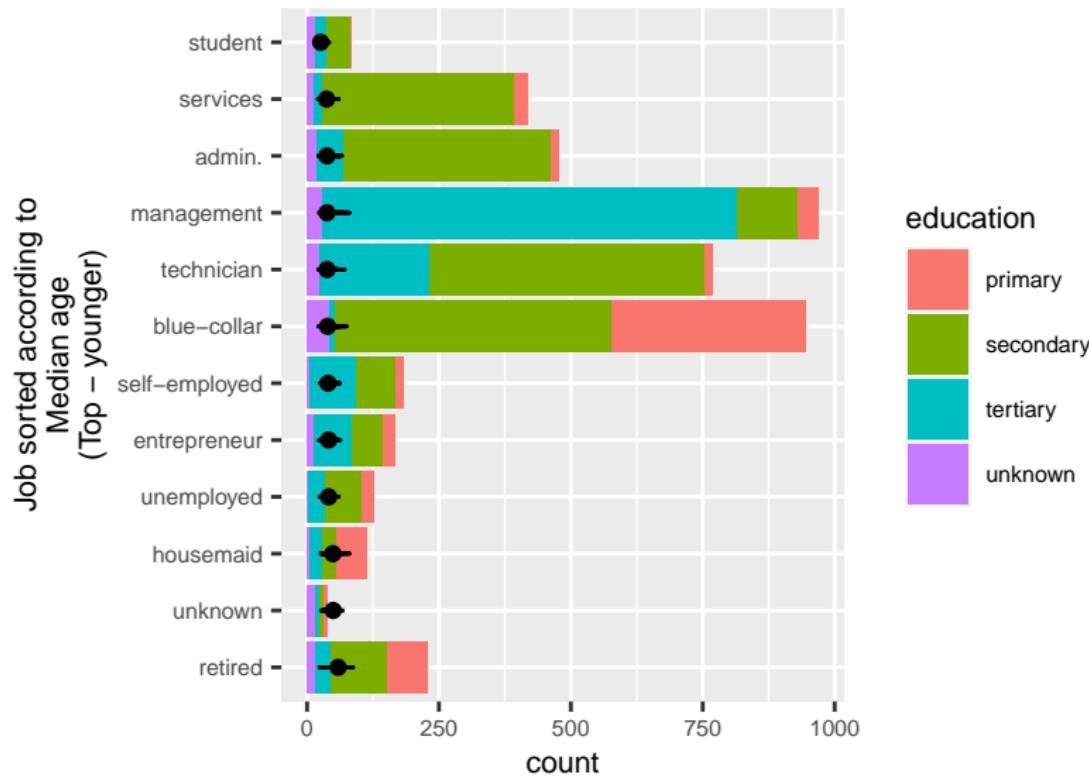
```
# If we want to order job according to alphabetical order.  
# use rev(levels(factor(bank$job)))  
ggplot(data = bank, mapping = aes(x = reorder(job, age, FUN = median),  
    fill = education)) + geom_bar() +  
  scale_x_discrete(limit = rev(levels(factor(bank$job)))) +  
  coord_flip() + theme(text = element_text(size=8))
```



Bar with composite data

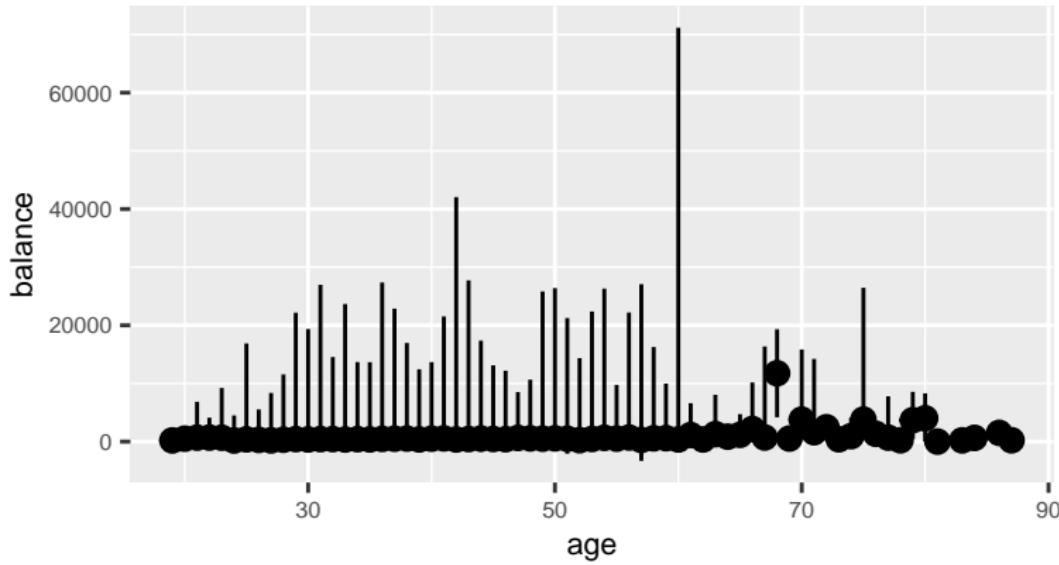
```
ggplot(data = bank, mapping = aes(x = reorder(job, age, FUN = median),  
    fill = education)) +  
# layer 1  
geom_bar() +  
# If we want to sort the job according to median age  
scale_x_discrete(limit =  
    rev(levels(reorder(bank$job, bank$age, FUN = median)))) +  
# And also add age range and median age.  
geom_line(aes(x = job, y = age)) +  
geom_point(data = group_by(bank, job) %>%  
    summarize(age = median(age)) %>% ungroup,  
    aes(x = job, y = age), inherit.aes = FALSE) +  
xlab("Job sorted according to\nMedian age\n(Top - younger)") +  
coord_flip() + theme(text = element_text(size=8))
```

Bar with composite data: plot



Data with statistical

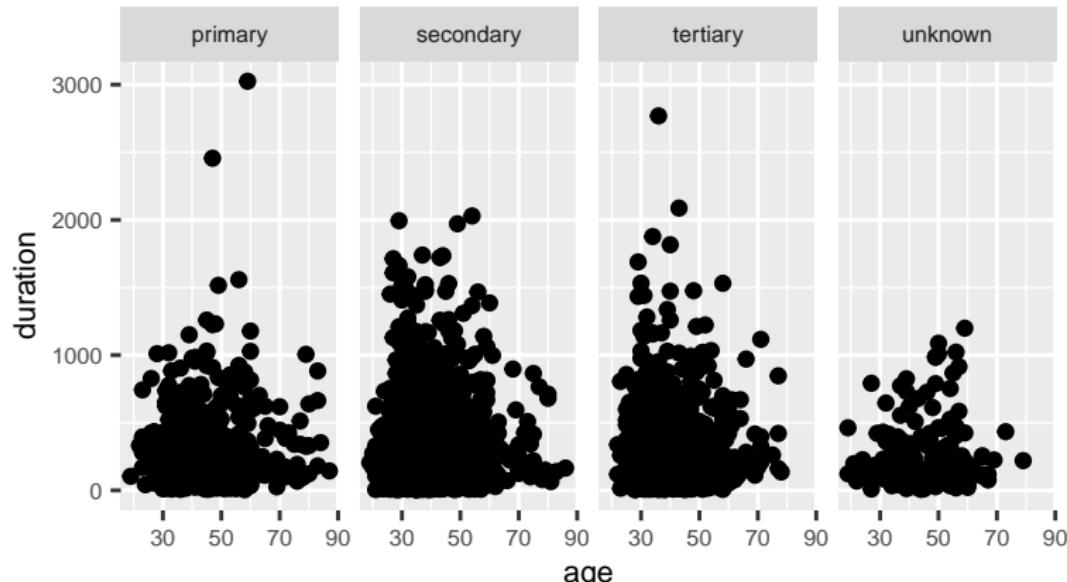
```
suppressWarnings({  
  ggplot(data = bank) +  
  stat_summary(  
    mapping = aes(x = age, y = balance),  
    fun.ymin = min,  
    fun.ymax = max,  
    fun.y = median  
  ) + theme(text = element_text(size=8))  
})
```



Facets

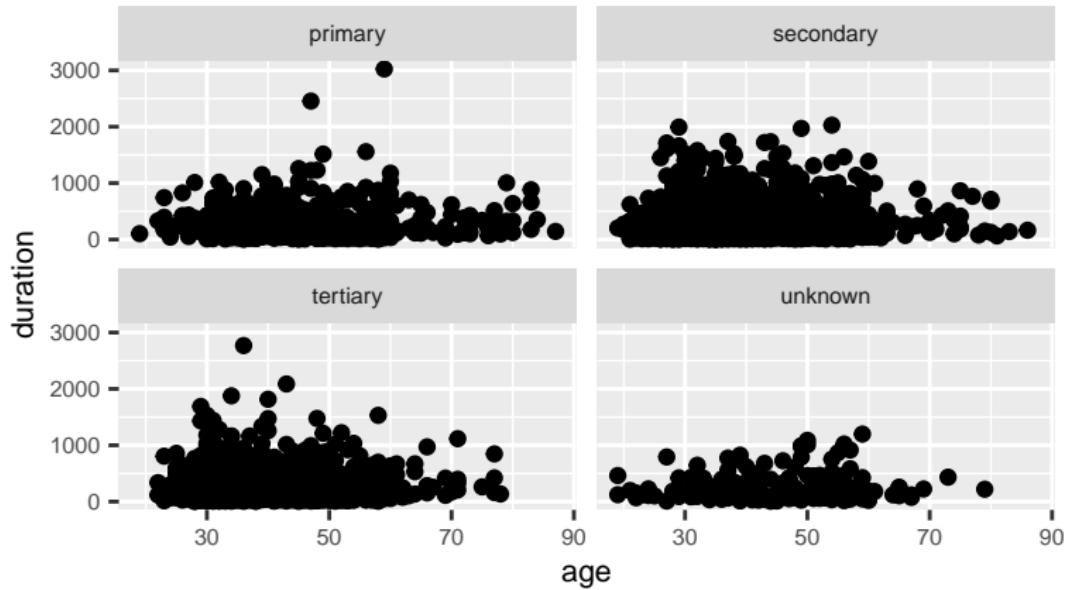
- Create individual figures.
- `facet_grid`: basic
- `facet_wrap`: you can control number of rows and cols

```
ggplot(data = bank) +  
  geom_point(mapping = aes(x = age, y = duration)) +  
  facet_grid(~ education) + theme(text = element_text(size=8))
```



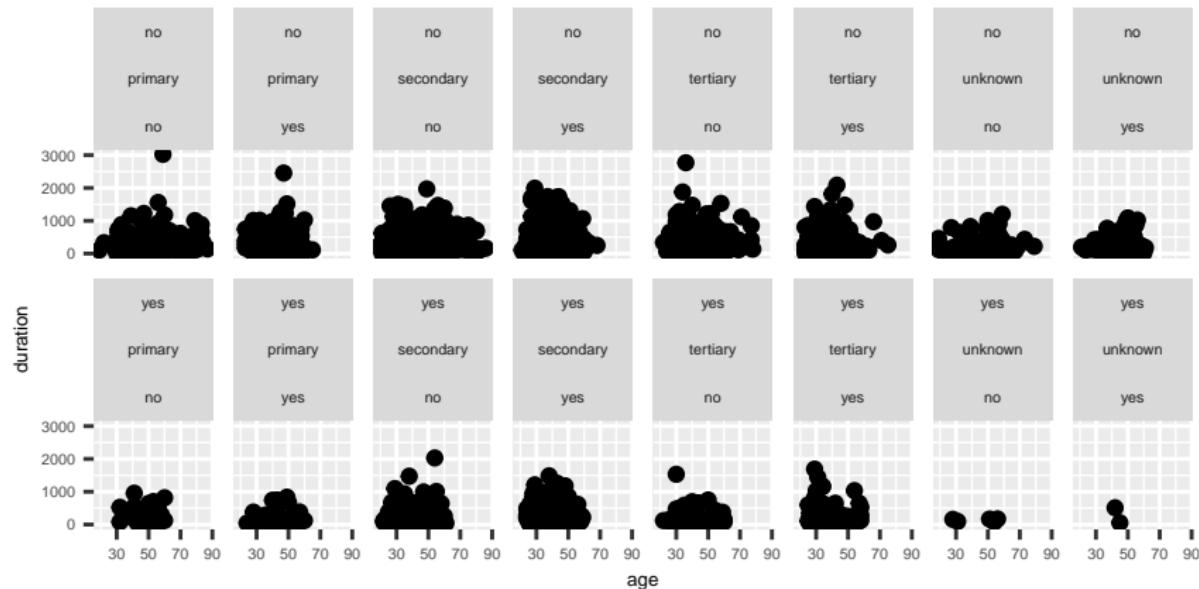
Facets - facet_wrap

```
ggplot(data = bank) +  
  geom_point(mapping = aes(x = age, y = duration)) +  
  facet_wrap(~ education, nrow = 2) +  
  theme(text = element_text(size=8))
```



Facets - facet_wrap multi-dimension

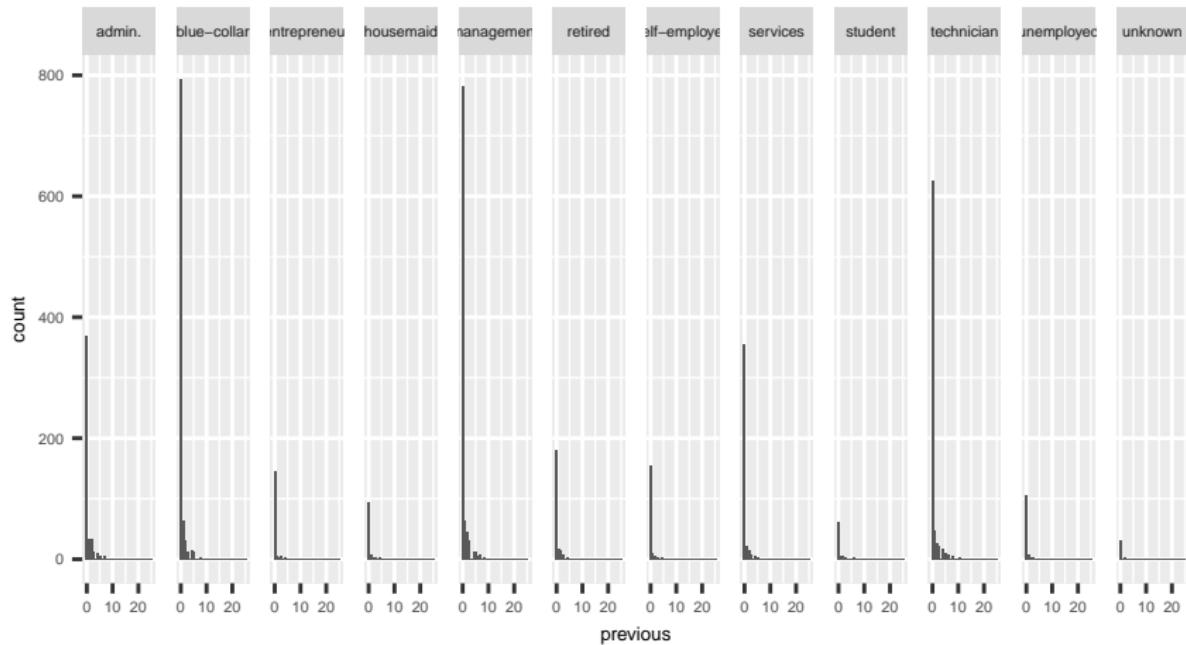
```
ggplot(data = bank) +  
  geom_point(mapping = aes(x = age, y = duration)) +  
  facet_wrap(loan ~ education ~ housing, nrow = 2) +  
  theme(text = element_text(size=5))
```



or we can use, *facet_grid(loan ~ education ~ housing)*

With facets or without facets? Case 1

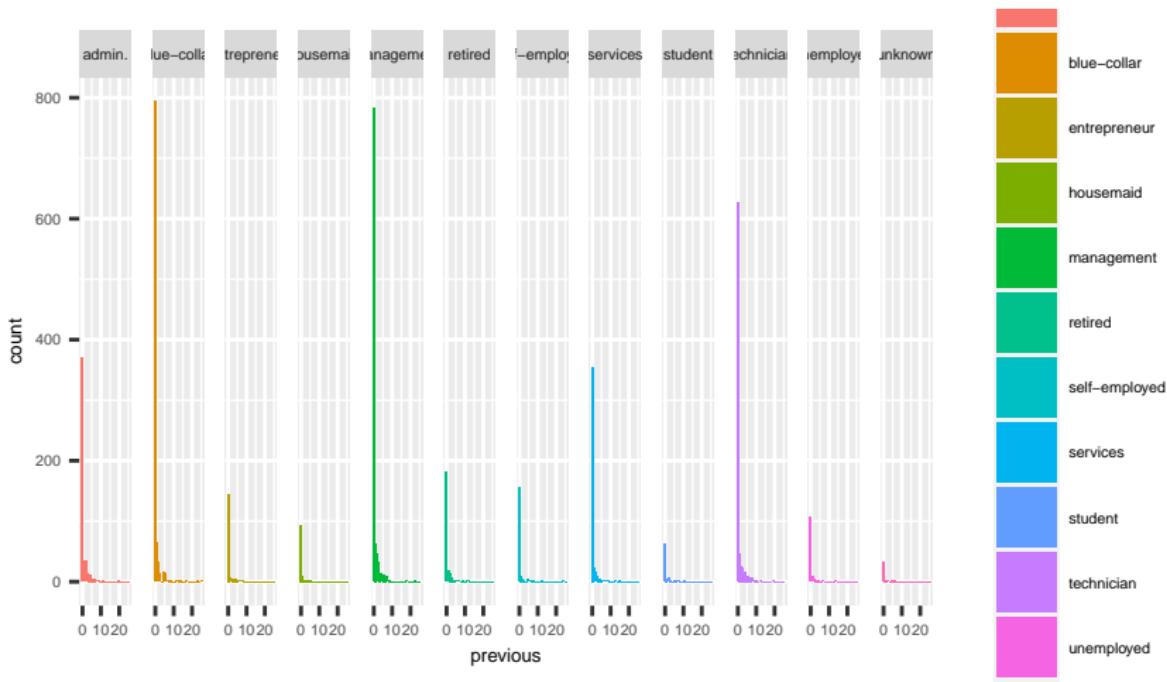
```
ggplot(bank, aes(previous)) + geom_histogram() + facet_grid(. ~ job) +  
  theme(text = element_text(size=5))  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



With facets or without facets? Case 1

- facets with color

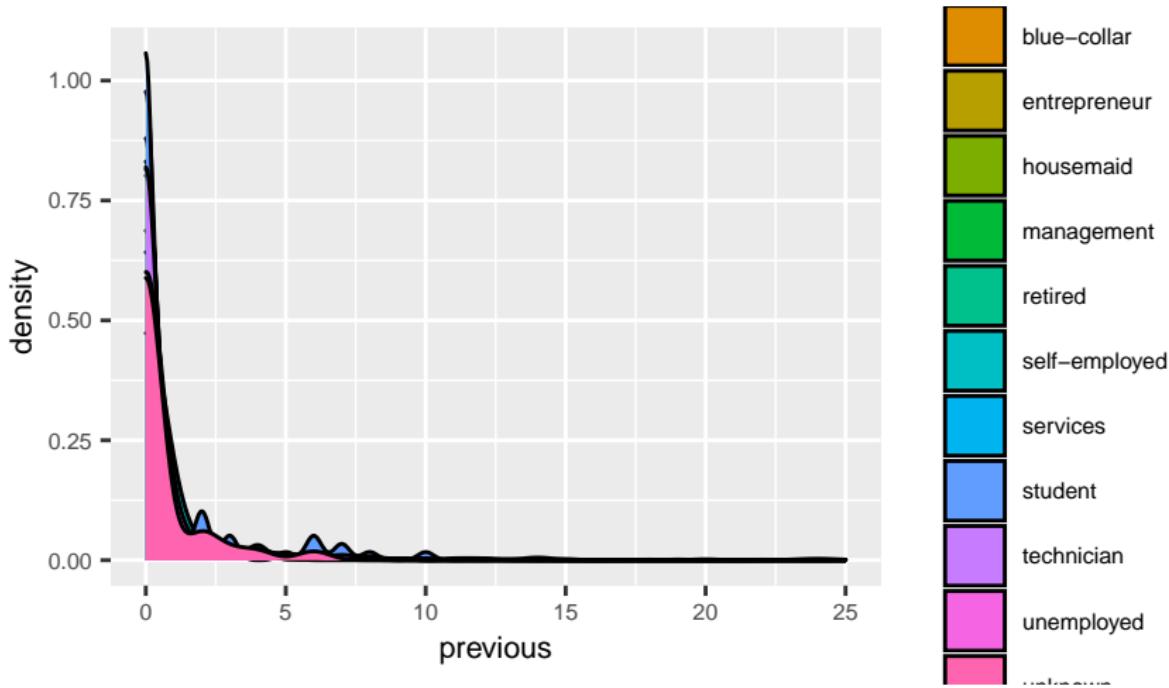
```
ggplot(bank, aes(previous)) + geom_histogram(aes(fill = job)) +  
  facet_grid(. ~ job) + theme(text = element_text(size=5))  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



With facets or without facets?

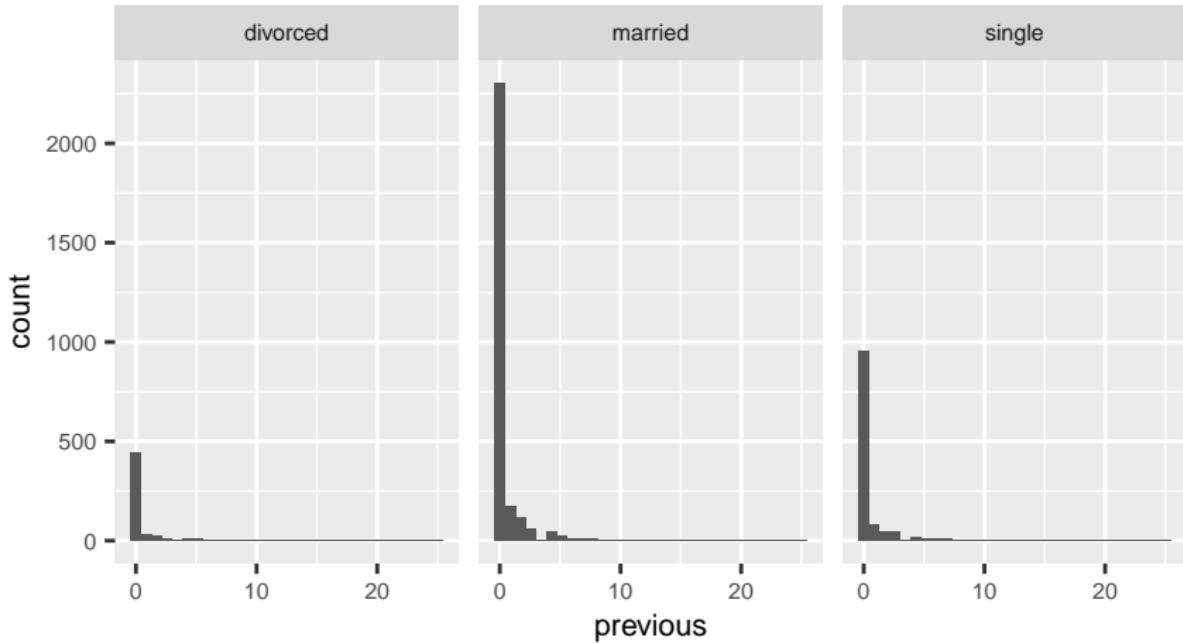
- Use density/color on one figure

```
ggplot(bank, aes(previous)) + geom_density(aes(fill = job)) +  
  theme(text = element_text(size=8))
```



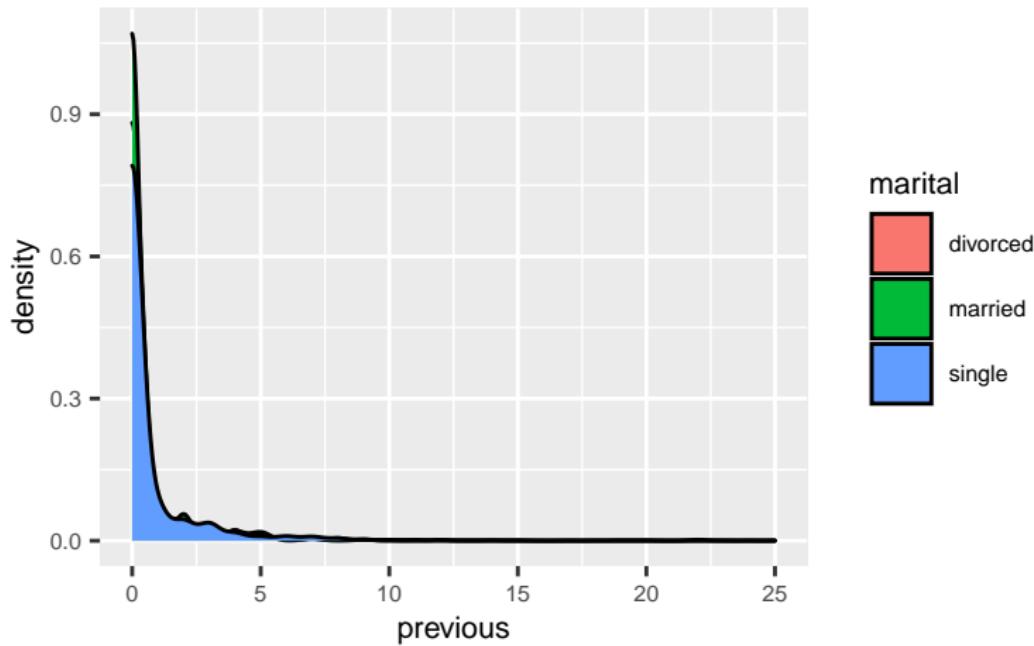
With facets or without facets: Case 2

```
ggplot(bank, aes(previous)) + geom_histogram() +  
  facet_grid(. ~ marital) + theme(text = element_text(size=8))  
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



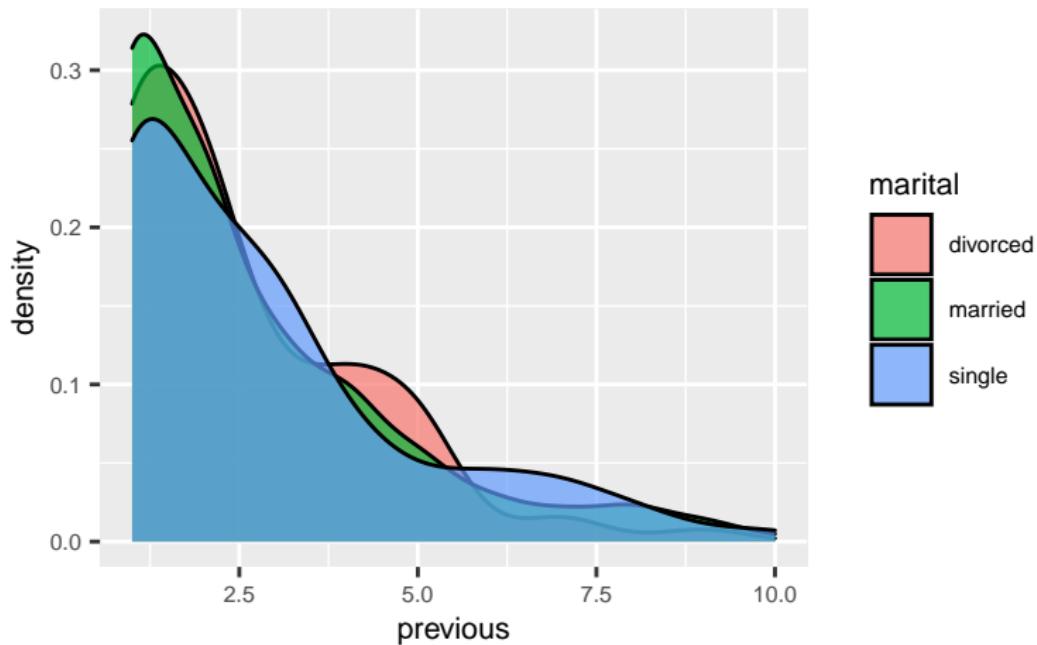
With facets or without facets: geom_density and fill

```
ggplot(bank, aes(previous)) + geom_density(aes(fill = marital)) +  
  theme(text = element_text(size=8))
```



With facets or without facets: xlim to zoom in

```
ggplot(bank, aes(previous)) +  
  geom_density(aes(fill = marital), alpha = 0.7) +  
  xlim(1, 10) + theme(text = element_text(size=8))  
## Warning: Removed 3725 rows containing non-finite values (stat_density).
```



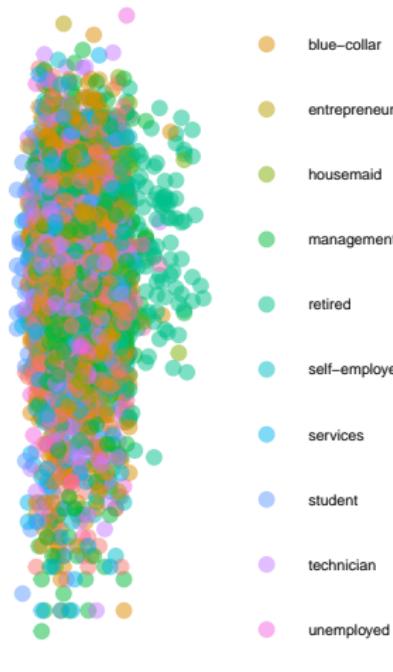
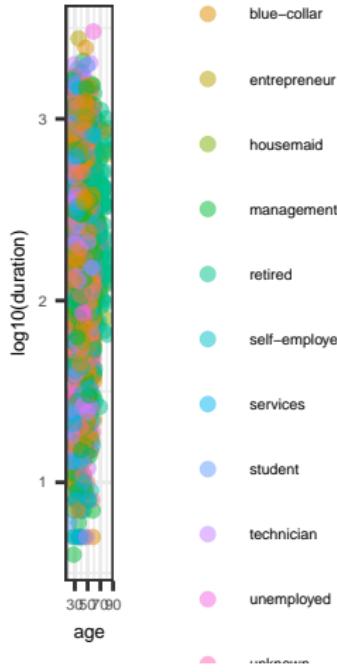
Add theme

The default theme is theme_gray()

```
g <- ggplot(bank, aes(x = age, y = log10(duration)))
g + geom_point(aes(color = job), size = 4, alpha = 1/2) + theme_bw()
g + geom_point(aes(color = job), size = 4, alpha = 1/2) + theme_void()

g + geom_point(aes(color = job), size = 4, alpha = 1/2) + theme_minimal() +
  labs(title = "Duration is longer with age",
       subtitle = "some random plot",
       caption = "from MFE") +
  labs(x = "age", y = expression("log " * Duration))
```

Add theme: result



ggthemes

- package ggthemes provides many other themes.

```
library(ggthemes)
## [1] "theme_base" "theme_calc"
## [3] "theme_economist" "theme_economist_white"
## [5] "theme_excel" "theme_few"
## [7] "theme_fivethirtyeight" "theme.foundation"
## [9] "theme_gdocs" "theme_hc"
## [11] "theme_igray" "theme_map"
## [13] "theme_pander" "theme_par"
## [15] "theme_solarized" "theme_solarized_2"
## [17] "theme_solid" "theme_stata"
## [19] "theme_tufte" "theme_wsj"
```

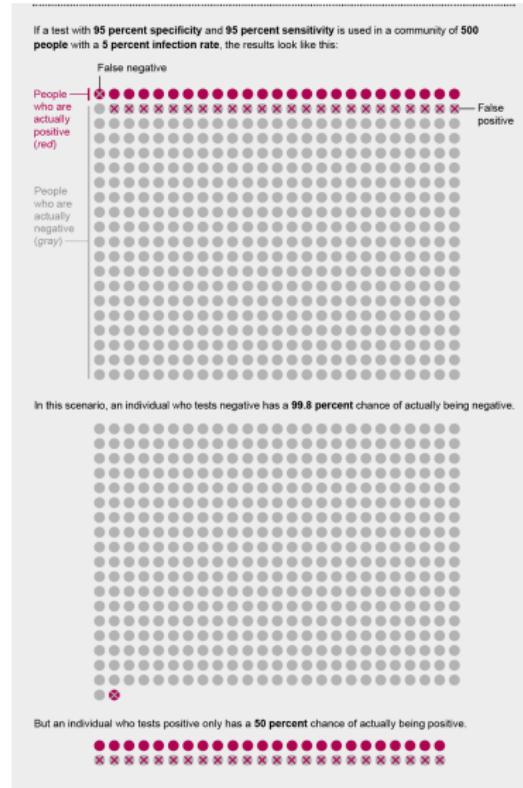
Take-home: ggplot summary

```
ggplot(data = <DATA>) +
<GEOM_FUNCTION>(
  mapping = aes(<MAPPINGS>),
  stat = <STAT>,
  position = <POSITION>
) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION>
```

Assignment

- In an R Markdown document, produce 6 data insights.
 - ▶ Each with a figure and a data story.
 - ▶ You can re-use up-to-3 existing examples.
- “Coronavirus Antibody Tests Have a Mathematical Pitfall”, title “False Positive Alarm” in Scientific American 323, 1, 12-13 (July 2020).
 - ▶ <https://www.scientificamerican.com/article/coronavirus-antibody-tests-have-a-mathematical-pitfall/>
 - ▶ Create a Shiny App that reproduces the infographic with input of infection rate.
 - ▶ Test specificity and sensitivity are fixed at 95%.

Assignment - 2

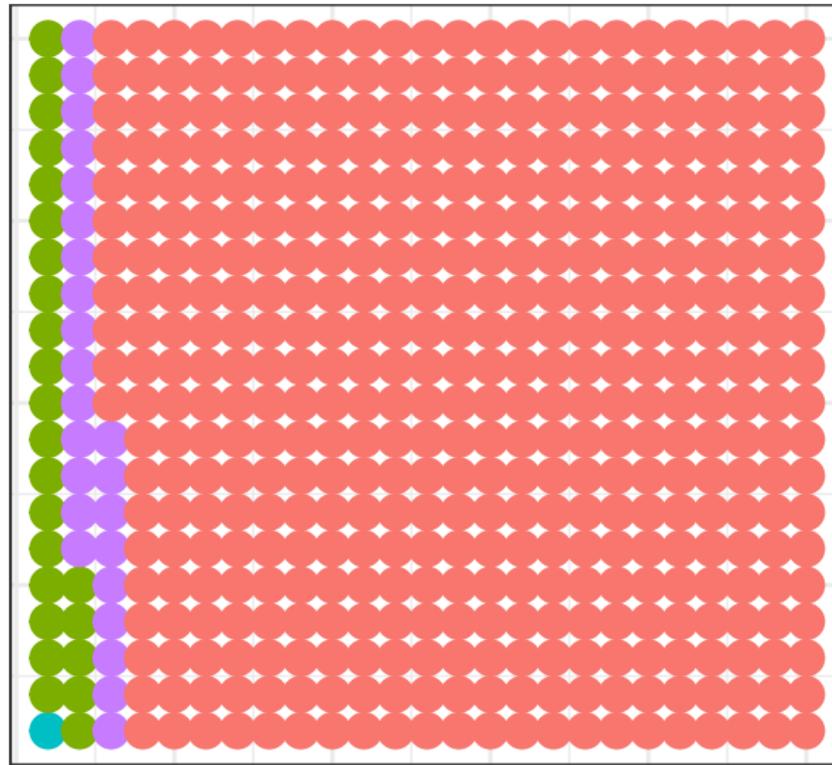


Assignment - 2

Below code to help start with.

```
df_sensi <- full_join(  
  tibble(x = 1:25, color = 'Actual Neg'),  
  tibble(y = 1:20, color = 'Actual Neg'),  
  by = 'color')  
  
# At 5% infection rate,  
# Positive 500 * 5% = 25 = Actual Pos (24 at 95%) + False Neg (1 at 5%)  
# Negative 500 * 95% = 475 = Actual Neg (451 at 95%) + False Pos (24 at 5%)  
df_sensi['color'] <- c(rep('False Neg', 1),  
                      rep('Actual Pos', 24),  
                      rep('False Pos', 24),  
                      rep('Actual Neg', 500 - 1 - 24 - 24))  
  
ggplot(df_sensi) +  
  geom_point(aes(x, y, colour = color),  
             size = 4, shape="circle") +  
  theme_bw() +  
  theme(axis.title.x=element_blank(), axis.title.y=element_blank(),  
        axis.line=element_blank(),axis.text.x=element_blank(),  
        axis.text.y=element_blank(),axis.ticks=element_blank())
```

Assgignment - 2



color

- Actual Neg
- Actual Pos
- False Neg
- False Pos

Assignment - 3 / 1

- Pick four stocks/ETFs + SPY (S&P 500 ETF) in US market. You may get the list of symbols here https://en.wikipedia.org/wiki/List_of_S%26P_500_companies
 - ▶ Download recent 100 day's price. E.g. `df <- av_get("MSFT", av_fun = "TIME_SERIES_DAILY_ADJUSTED")`
 - ▶ Save them into `/data` directory under `/Assignment 4`.
 - ▶ With \$1mio cash, invest with equal-weighted (long only) in the four stocks/ETF on the 1st day with `adjusted_close`.
 - ▶ Take cash reserve of about 10% (about 90% are investable). Make sure invest in integer number of shares.
 - ▶ We only use `adjusted_close` for buying/selling price and valuation.

Assignment - 3 / 2

- Suppose you are to report this investment to client. Do below in a R Markdown document
 - ▶ Load the data.
 - ▶ Decide the allocation n_i per stock. Calculate daily valuation of the portfolio,
$$Value = \sum n_i * price_i$$
, return of the portfolio, $R_p = \frac{Value_{d+1}}{Value_d} - 1$
 - ▶ Calculate the Sharpe ratio for this portfolio and S&P: $Sharpe\ Ratio = \frac{R_p(\text{mean}) - R_f}{\sigma_p}$ with $R_f = 0.01/250$, $\sigma_p = Stdev(R_p)$
 - ▶ Plot a histogram of daily return.
 - ▶ Plot total valuation, and relativity return v.s time for your investment portfolio and S&P.
 - ★ The total valuations and relative return for S&P is as if you would have invested \$1mio cash into SPY instead (still with 10% cash and integer number of shares)
 - ▶ Plot relative valuation for the different allocation (stocks and cash)
 - ★ Hint: similar output as `geom_bar(aes(...), position = 'full')`. `geom_bar` is for discrete variable. `geom_area(aes(x,y,color),position='full')` is for continuous/numeric variable.