ggplot2

Youmi Suk

School of Data Science, University of Virginia

1. 24. 2022

ggplot2

- Decompose graphics into eight components:
    - Data
    - Mapping
    - Geometries
    - Statistics
    - Scales
    - Facets
    - Coordinates
    - Themes

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

ggplot2: Data, Mapping, and Geometries

- First, you need data!
- Mapping allows datasets to be understood by the graphic system.
    - *Aesthetic* mapping: link variables in data to graphical properties in the geometry.
- Geometries imply how to interpret aesthetics as graphical representations, and determine your plot type, e.g., `geom_point`, `geom_line`, `geom_bar`.

ggplot2: Statistics

- Statistics transform input variables to displayed values.
  - Count numbers of observations in each category for a bar chart.
  - Calculate summary statistics for a boxplot.
- Essential statistics for plotting are computed by default.

ggplot2: Scales

- Scales imply a specific interpretation of values (discrete, continuous, etc.)
- Everything inside `aes()` will have a scale. If none is provided it will get a default.
- Scales follow a predictable naming scheme: `scale_<aesthetic>_<type>()`
- `<type>` can either be generic (e.g., continuous, discrete, or binned) or specific (e.g., log transformation, area)

ggplot2: Facets

- Facets split data into multiple panels.
- Facets should not be used to combine multiple separate plots.
- `ggplot2` provides two facets for splitting data by categories: `facet_wrap()` and `facet_grid()`.

ggplot2: Coordinates

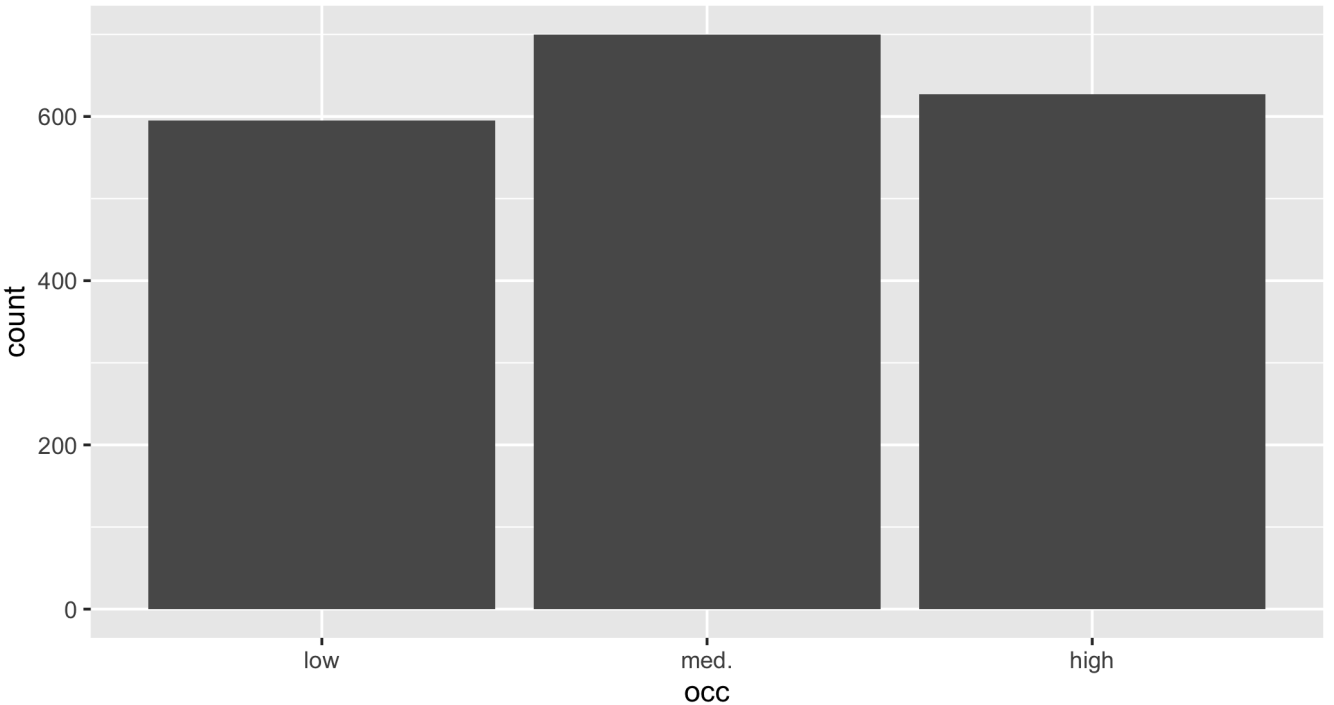
- How should x or y be interpreted?
- Limits and transformation can be applied in scale or in coord.
- Extremely useful in cartography (map projections)

ggplot2: Themes

- Stylistic changes to the plot not related to data.
- Themes can both apply complete themes or modify elements directly.

Barplot

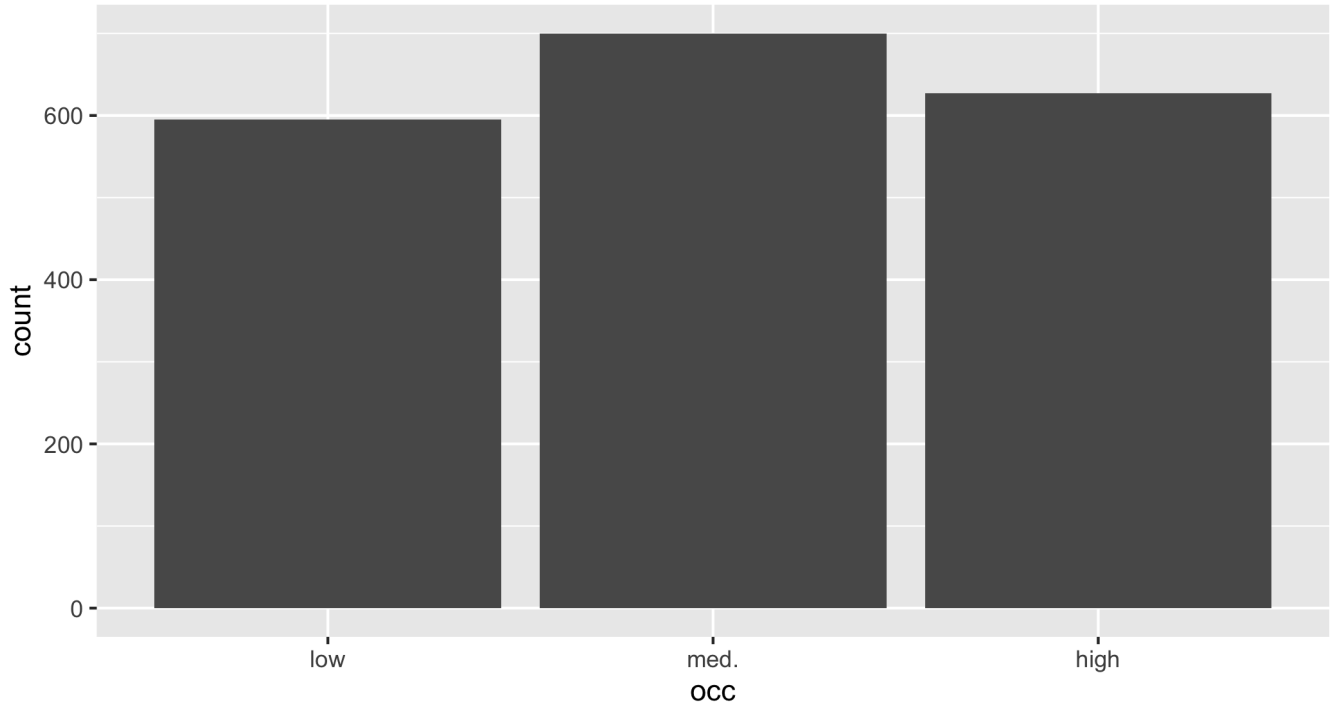
```
ggplot(data=incex) + geom_bar(mapping = aes(x = occ))
```



Barplot

- `geom_bar()` uses `stat_count()` by default. We can reproduce the barplot using `stat_count()`.

```
ggplot(incex) + stat_count(aes(x = occ))
```



Barplot

You may have precomputed data.

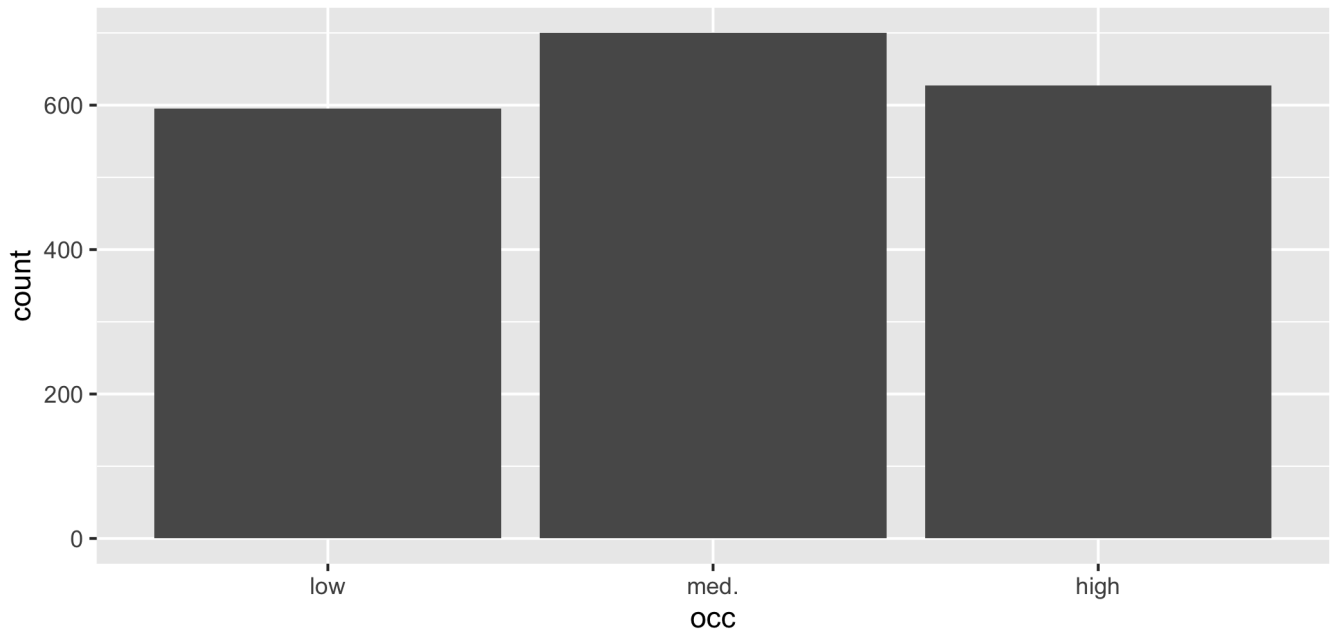
```
occ_counted <- incex %>% count(occ, name = 'count')
occ_counted
```

```
##    occ count
## 1  low   595
## 2 med.   700
## 3 high   627
```

Barplot

- With precomputed data, use `identity` stat.

```
ggplot(occ_counted) + geom_bar(aes(x = occ, y = count),
                               stat = 'identity')
```

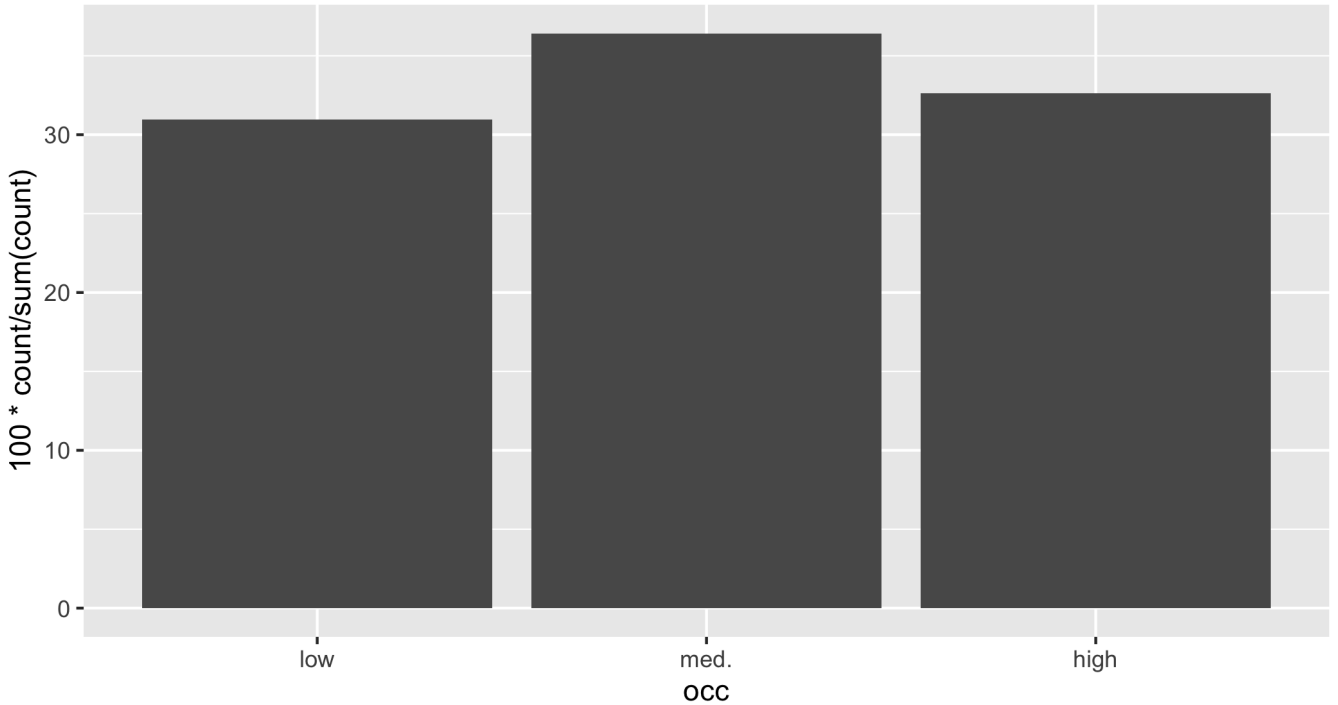



```
ggplot(occ_counted) + geom_col(aes(x = occ, y = count)) # shortcut
```

Barplot

- Use `after_stat()` to modify mapping from stats (here we calculate %).

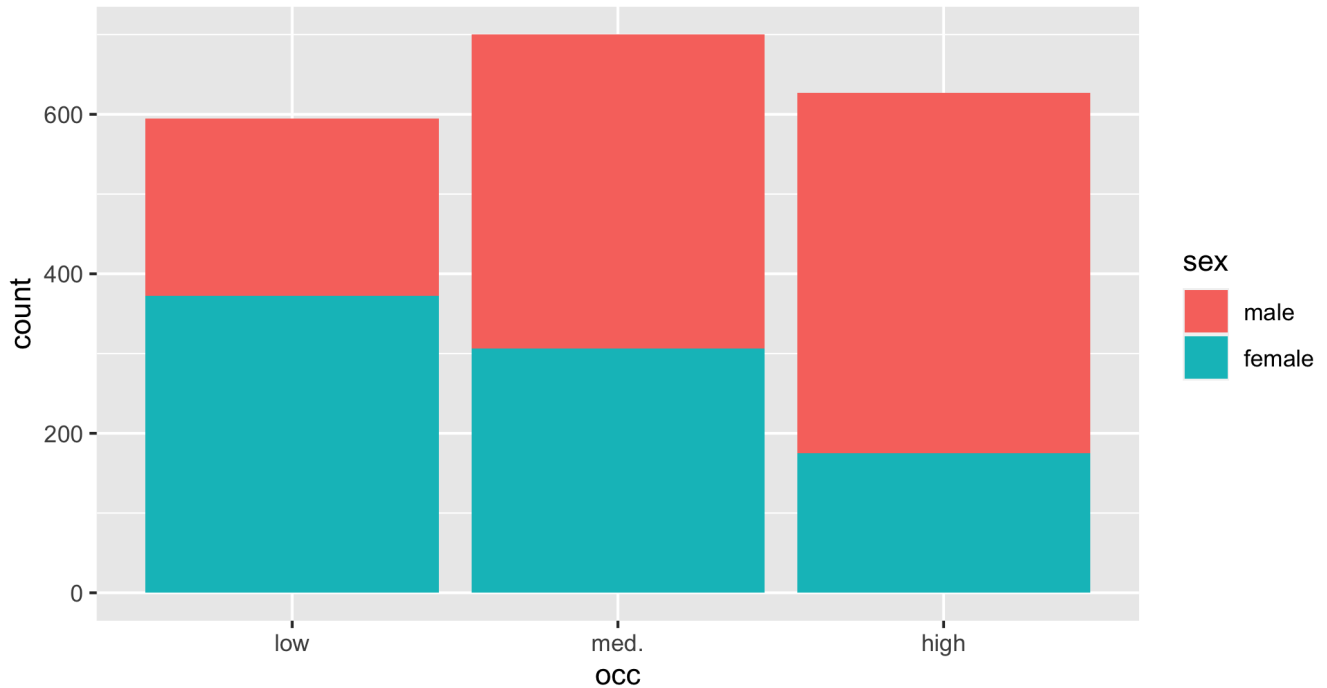
```
ggplot(incex) + geom_bar(aes(x = occ,
  y = after_stat(100 * count / sum(count))))
```



Barplot

- A stacked plot with frequencies.
- Based on the vector type of the variable *sex*, a discrete colour scale is picked.

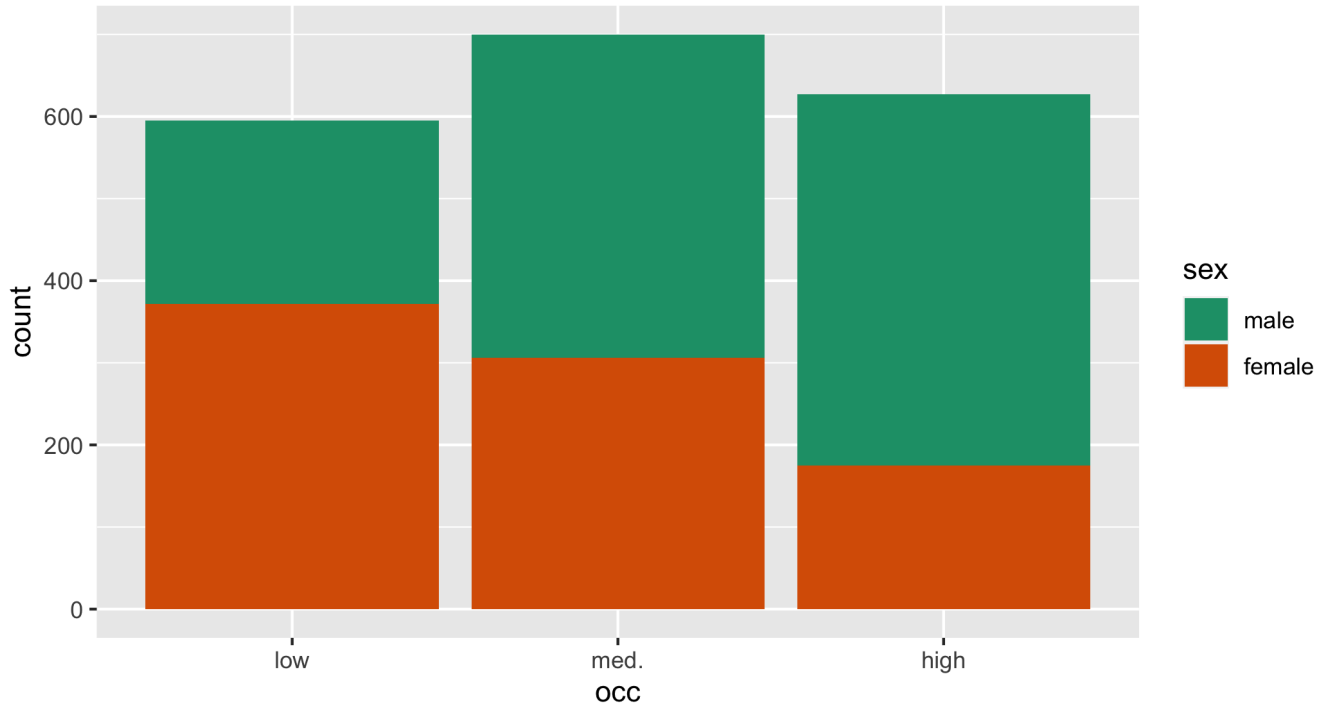
```
ggplot(incex) + geom_bar(aes(x = occ, fill=sex))
```



Barplot

- You may want to change colors.

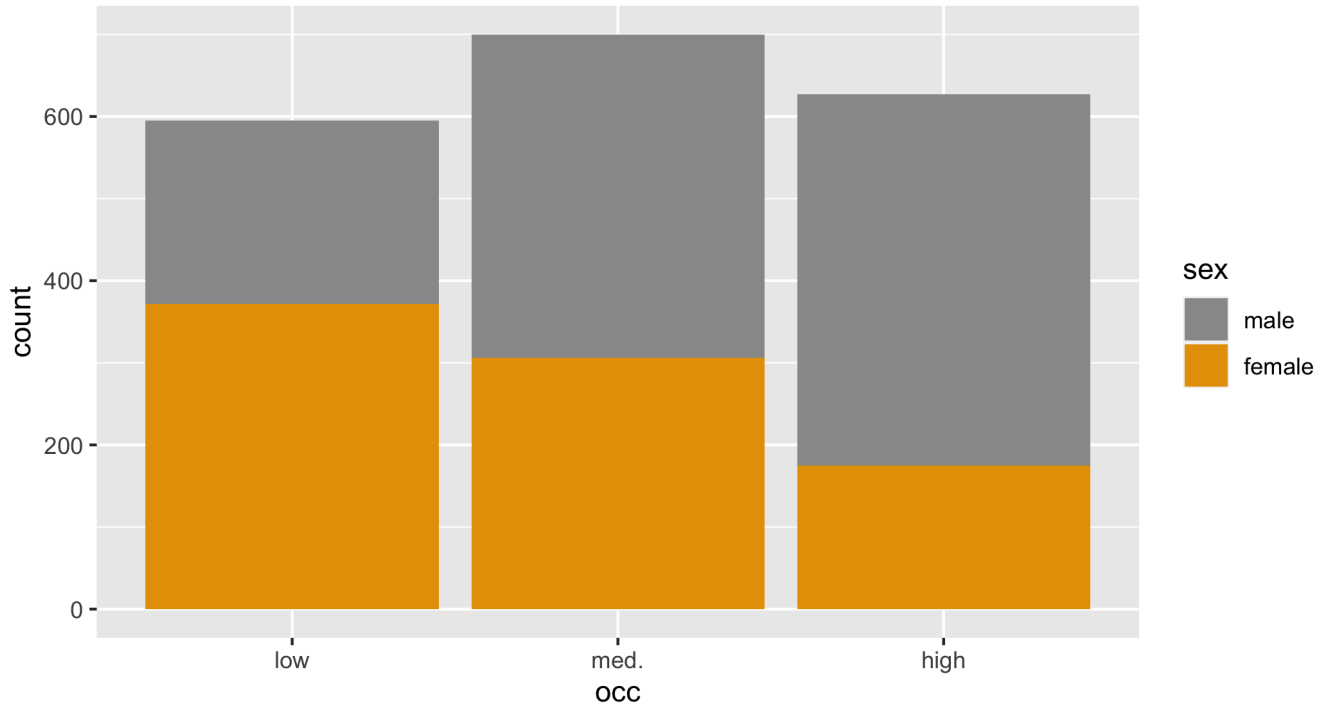
```
ggplot(incex) + geom_bar(aes(x = occ, fill=sex)) +
  scale_fill_brewer(palette="Dark2")
```



Barplot

- You may want to choose colors manually.

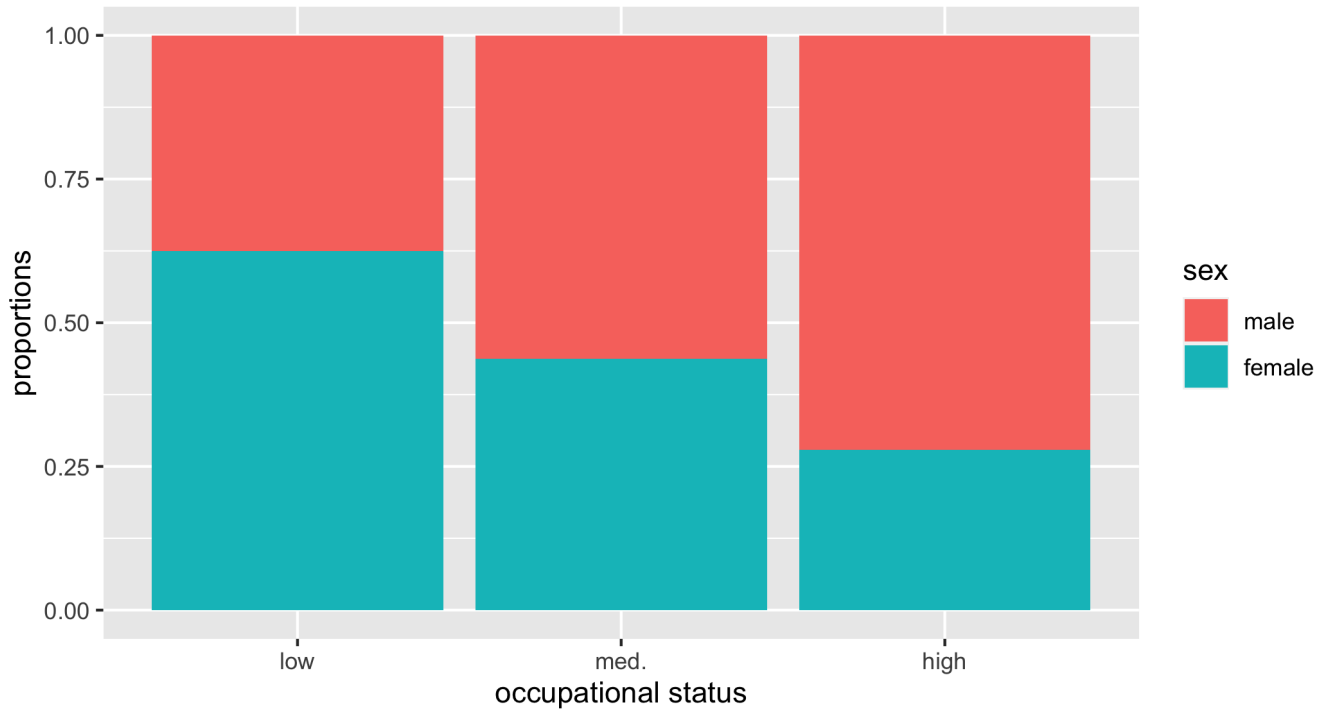
```
ggplot(incex) + geom_bar(aes(x = occ, fill=sex)) +
  scale_fill_manual(values=c("#999999", "#E69F00"))
```



Barplot

- A stacked plot with proportions. `position = "fill"` makes “proportion bars”.

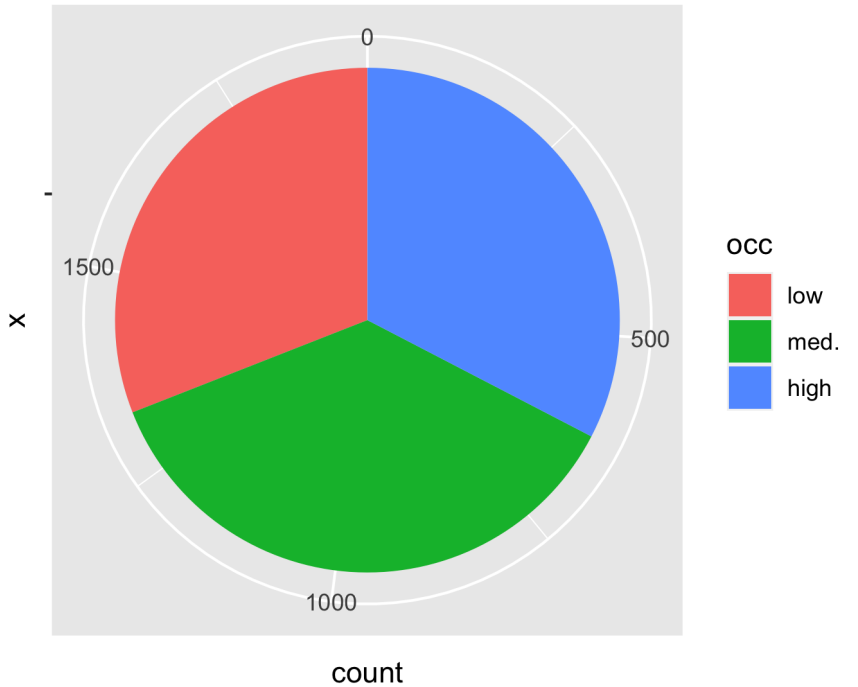
```
ggplot(incex) + geom_bar(aes(x = occ, fill=sex), position = "fill") +
  labs(x="occupational status", y="proportions")
```



Pie Chart

- A polar coordinate system interprets `x` and `y` as radius and angle.

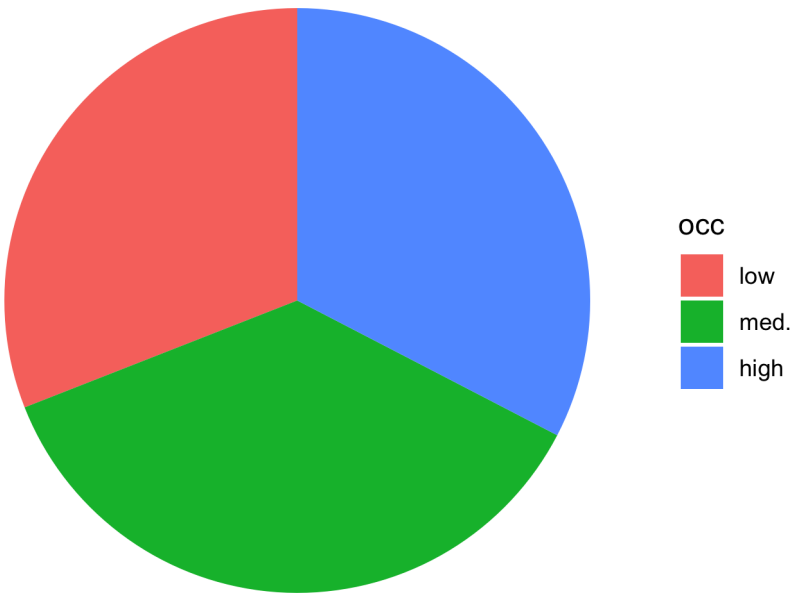
```
occ_counted <- incex %>% count(occ, name = 'count')
p <- ggplot(occ_counted) + geom_bar(aes(x="", y = count, fill=occ),
                                    stat = 'identity', width=1)
p + coord_polar(theta = "y", start=0)
```



Pie Chart

- Let's change `theme`. Here, `theme_void()` removes background, grid, numeric labels.

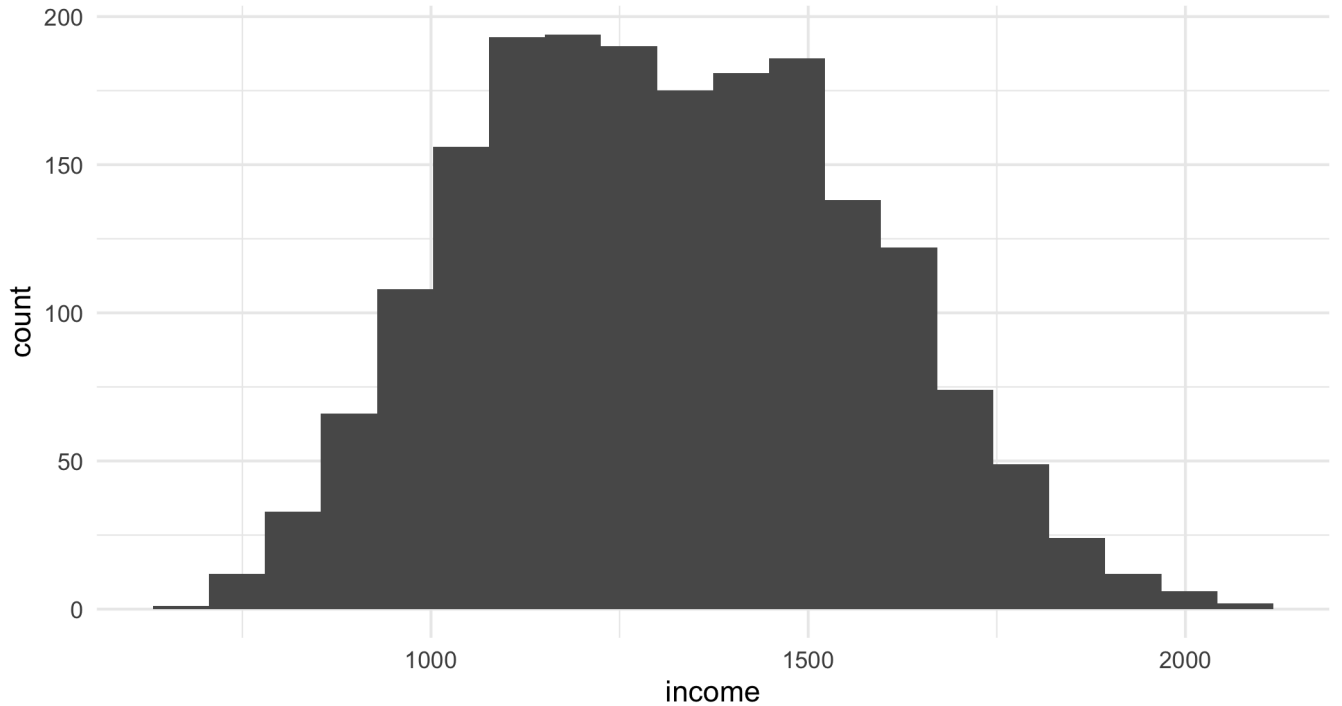
```
p + coord_polar(theta = "y", start=0) +
  theme_void()
```



Histogram

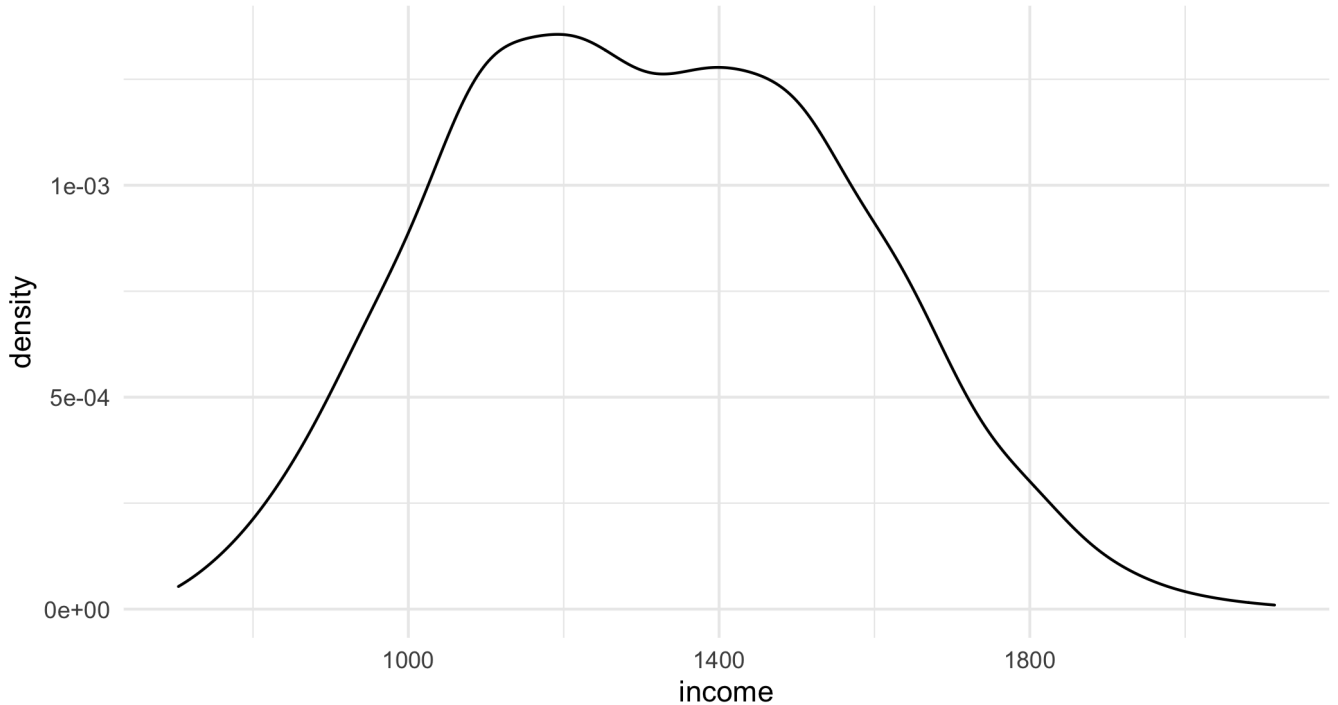
- Draw a histogram with a continuous variable.

```
ggplot(incex) + geom_histogram(aes(x = income), bins = 20) + theme_m
```



Density Plot

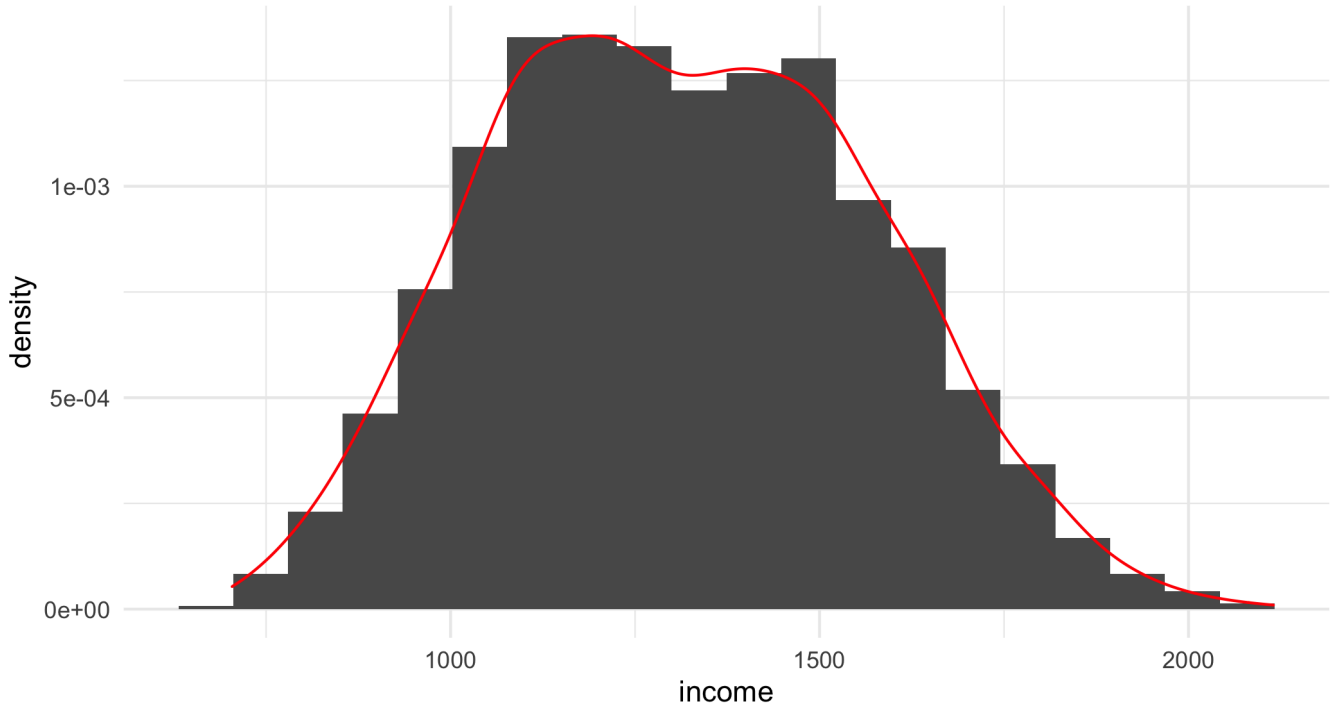
```
ggplot(incex) + geom_density(aes(x = income)) + theme_minimal()
```



Histogram & Density Plot

- Draw the histogram and density plots together.

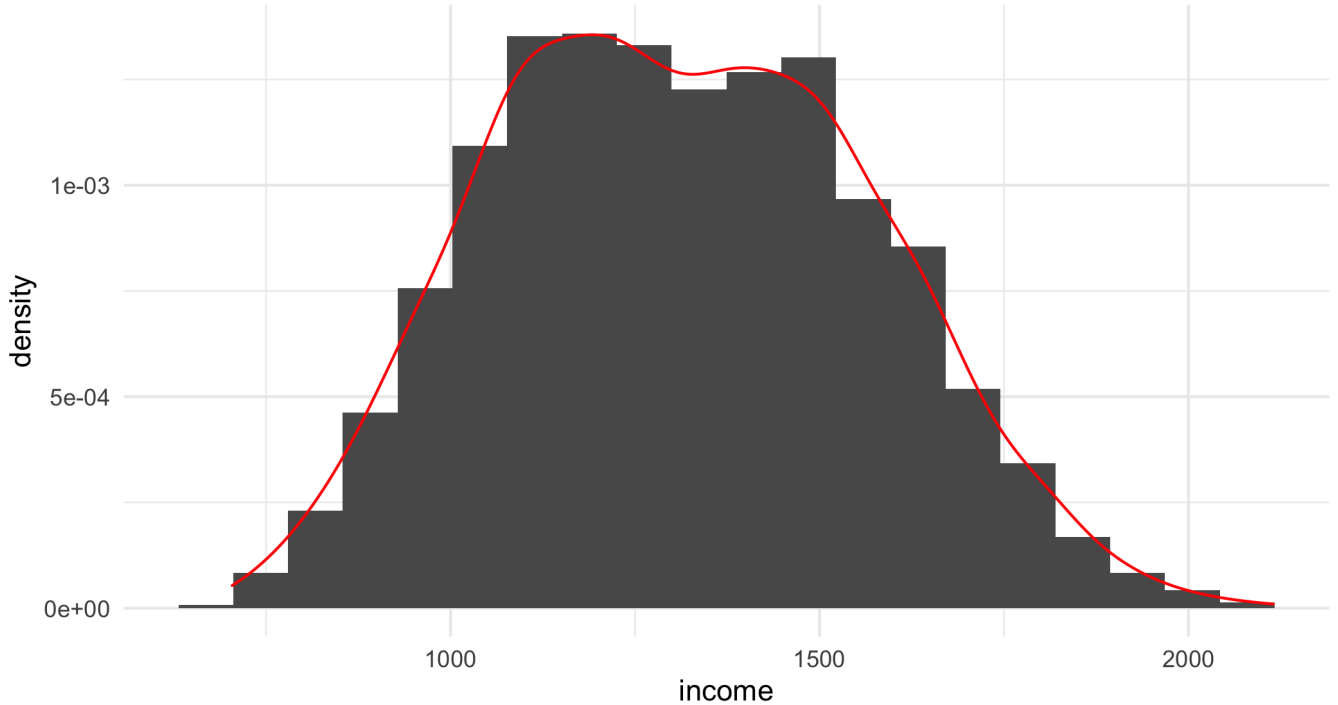
```
ggplot(incex) + geom_histogram(aes(x = income, y=..density..), bins=2
  geom_density(aes(x = income), col="red") + theme_minimal()
```



Histogram & Density Plot

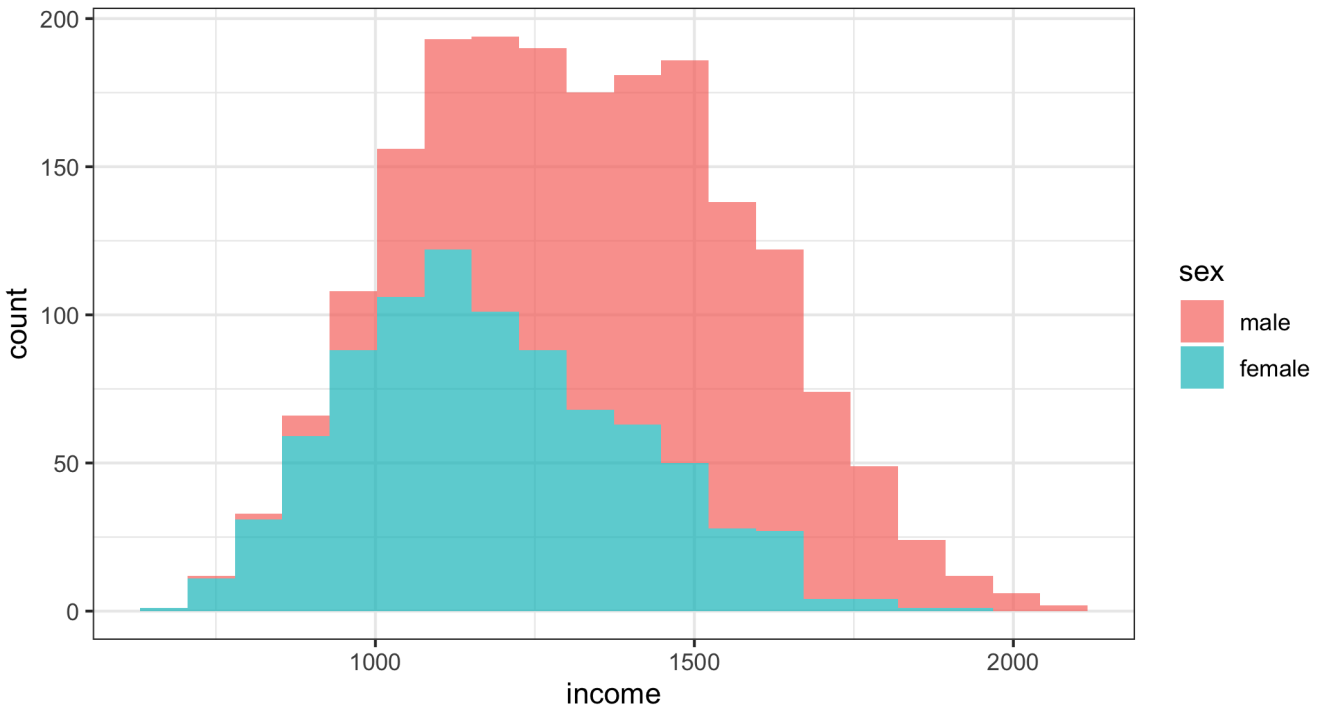
- We can globally define mapping parameters inside `ggplot`.

```
ggplot(incex, aes(x = income, y=..density..)) +
  geom_histogram(bins=20) + geom_density(col="red") + theme_minimal()
```



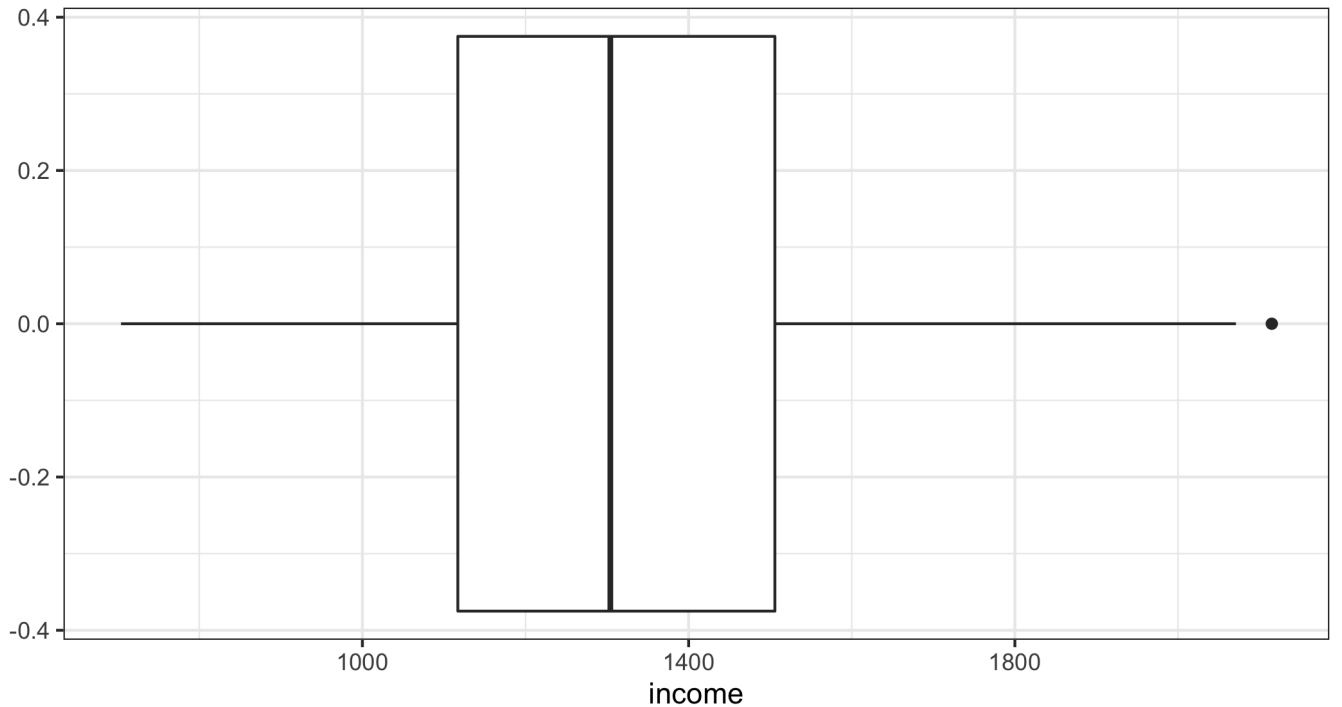
Histogram with Groups

```
ggplot(incex) + geom_histogram(aes(x = income, fill=sex),
  bins=20, alpha=0.7) + theme_bw()
```



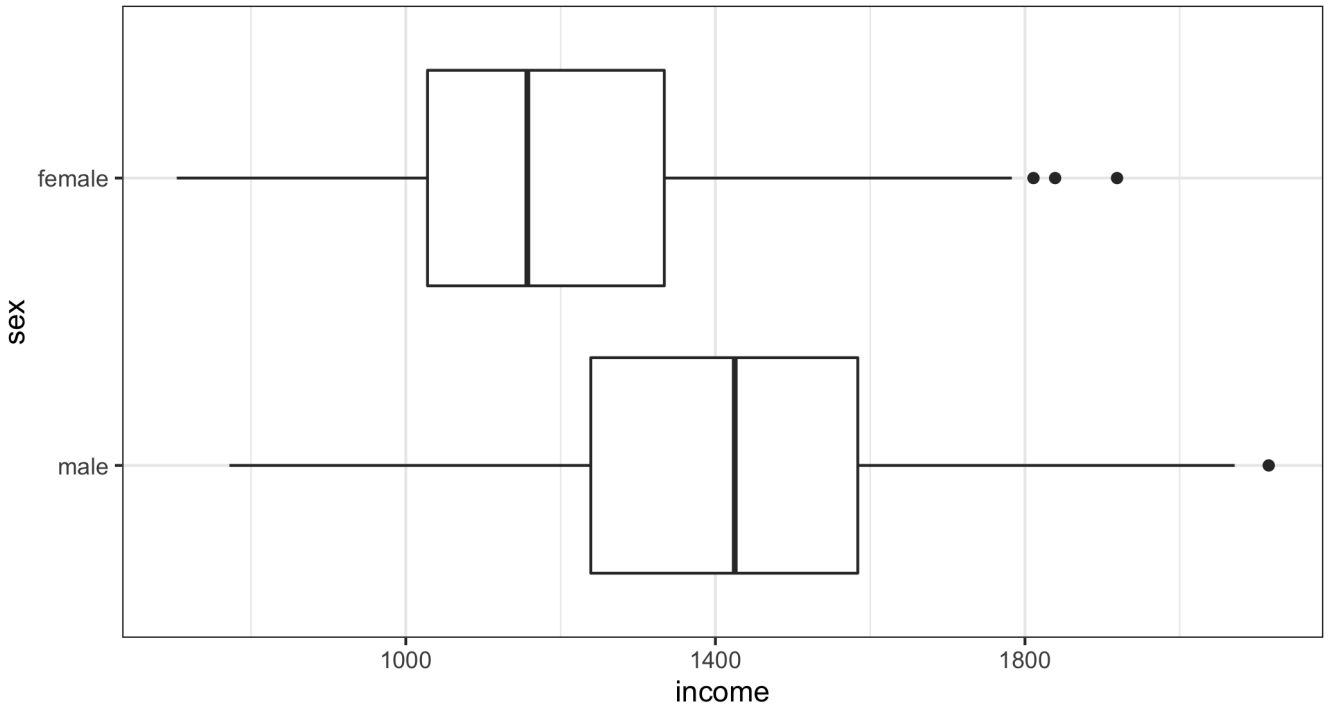
Boxplot

```
ggplot(incex, aes(x = income)) + geom_boxplot() + theme_bw()
```



Boxplot

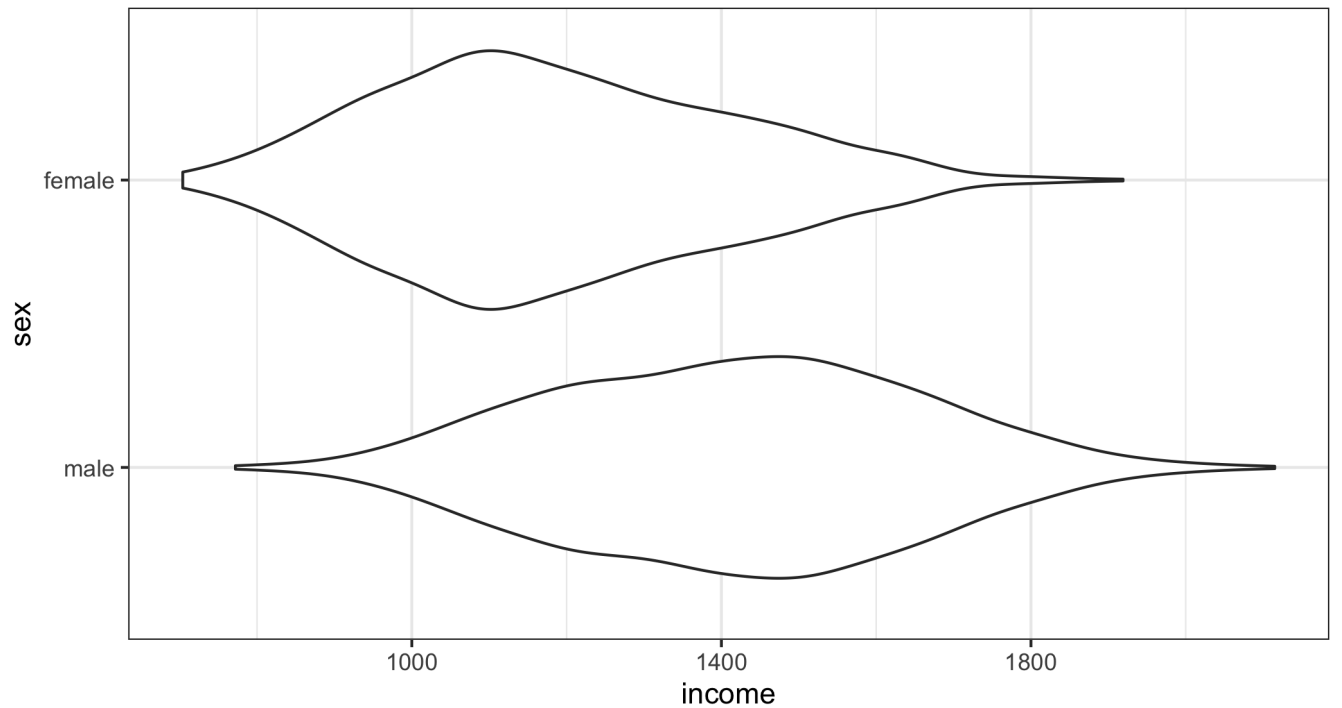
```
ggplot(incex, aes(x = income, y = sex)) + geom_boxplot() + theme_bw()
```



Violin Plot

- A violin plot displays distributions of numeric data for one or more groups using density curves. The width of each curve corresponds with the approximate frequency of data points in each region.

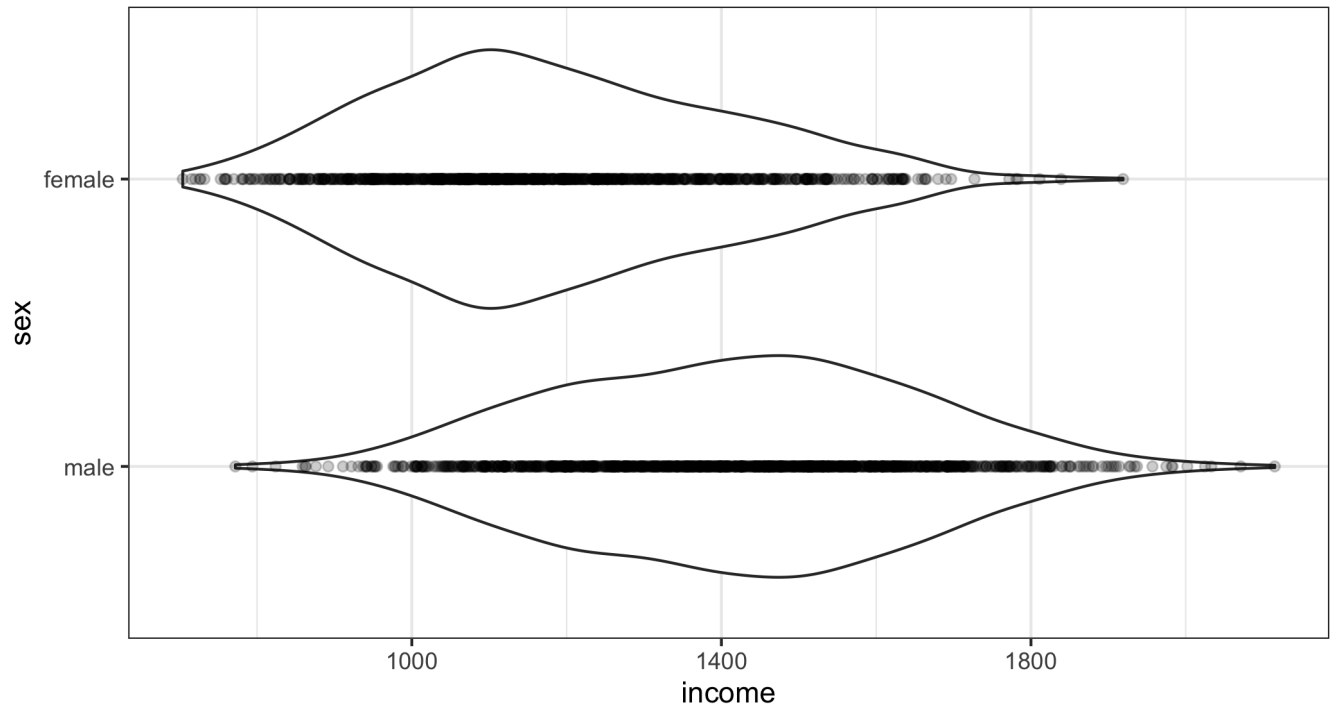
```
ggplot(incex, aes(x = income, y = sex)) + geom_violin() + theme_bw()
```



Violin Plot

- You may want to overlay data points.

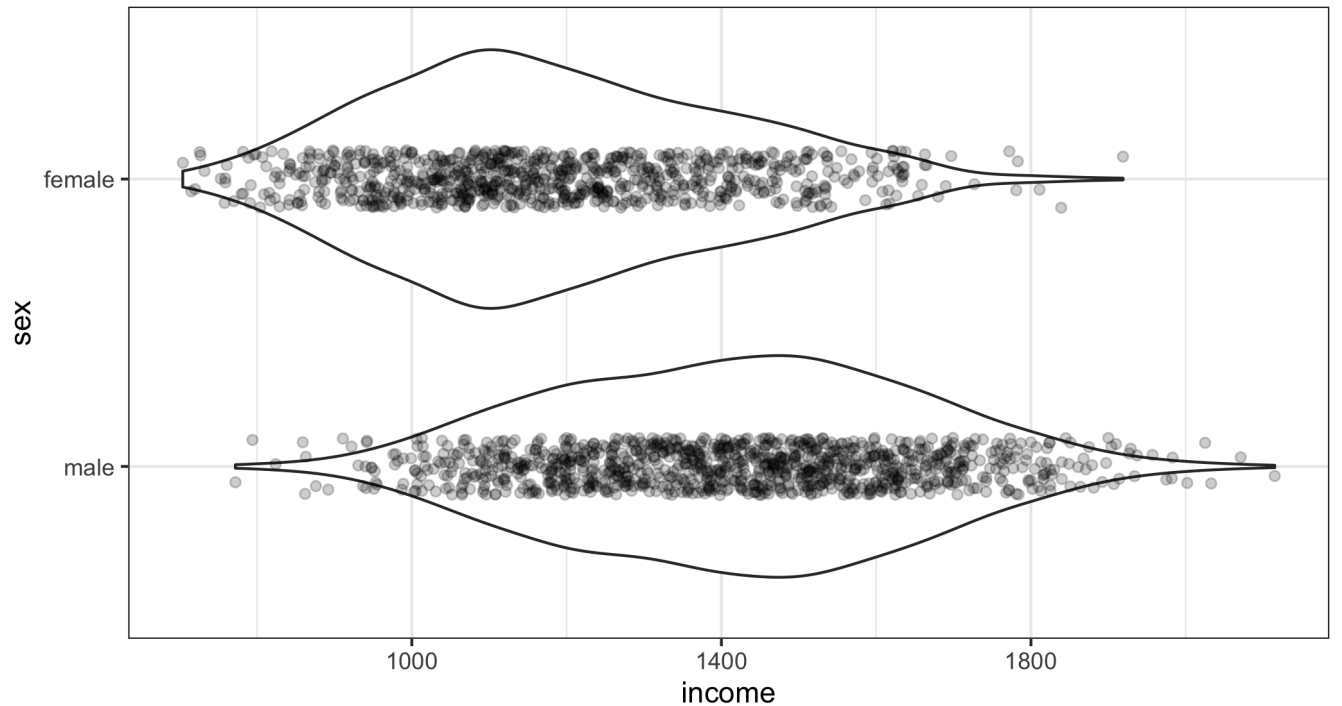
```
ggplot(incex, aes(x = income, y = sex)) + geom_violin() +
  geom_point(alpha=0.2) + theme_bw()
```



Violin Plot

- You may want to jitter data points.

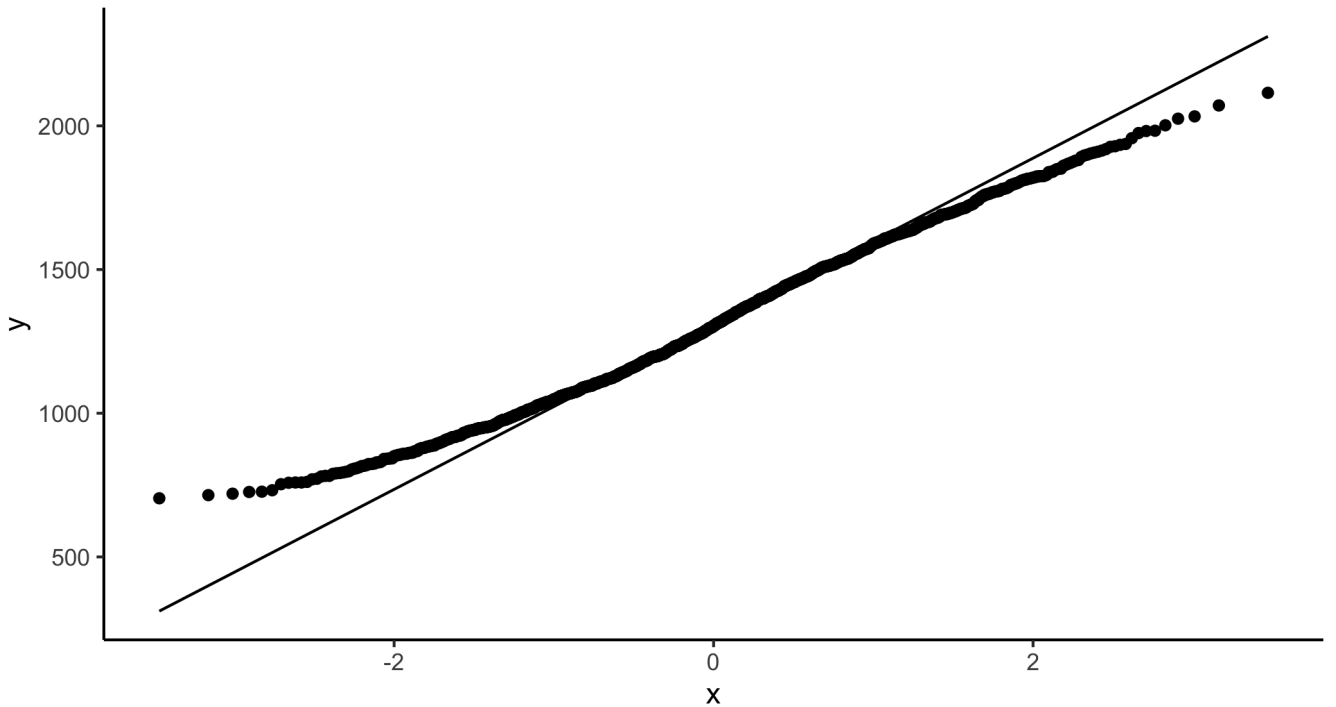
```
ggplot(incex, aes(x = income, y = sex)) + geom_violin() +
  geom_jitter(width = 0.1, height = 0.1, alpha=0.2) + theme_bw()
```



Quantile-Comparison Plot

- Compare the observed (empirical) sample distribution of a variable with a theoretical distribution.

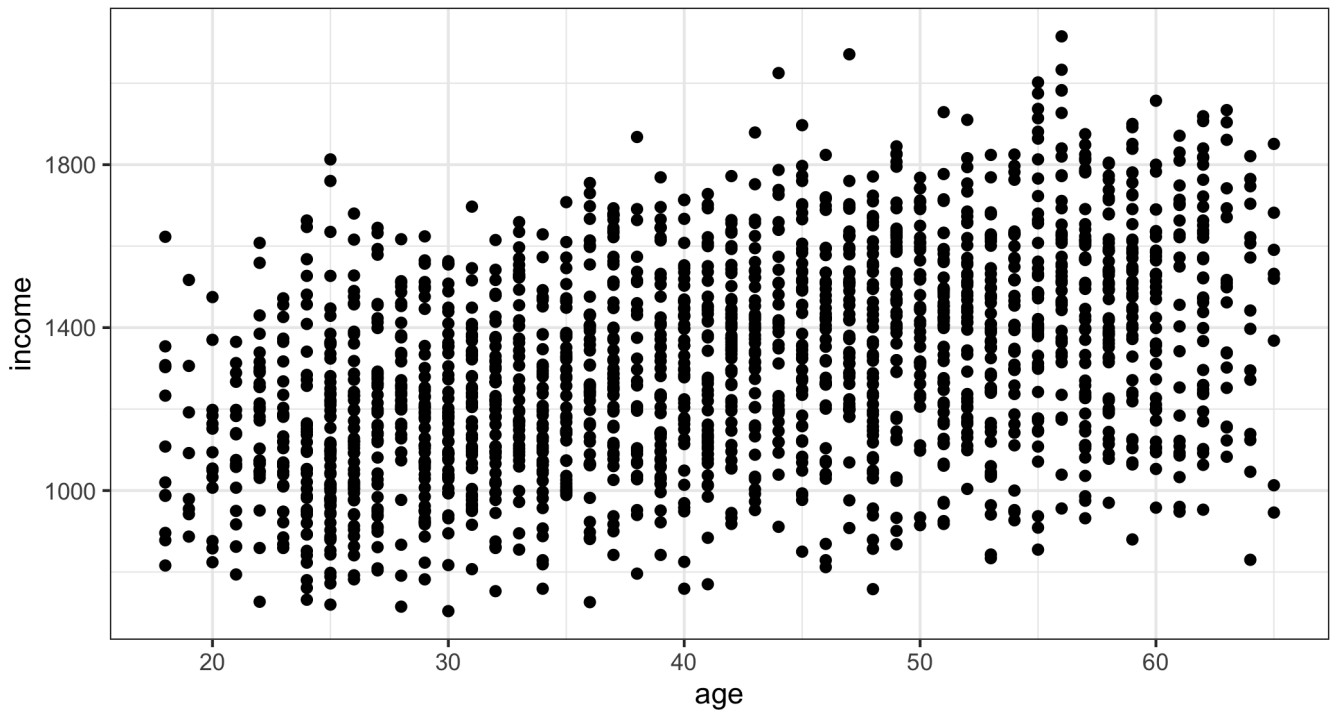
```
ggplot(incex, aes(sample = income)) + stat_qq() + stat_qq_line() + th
```



Scatterplot

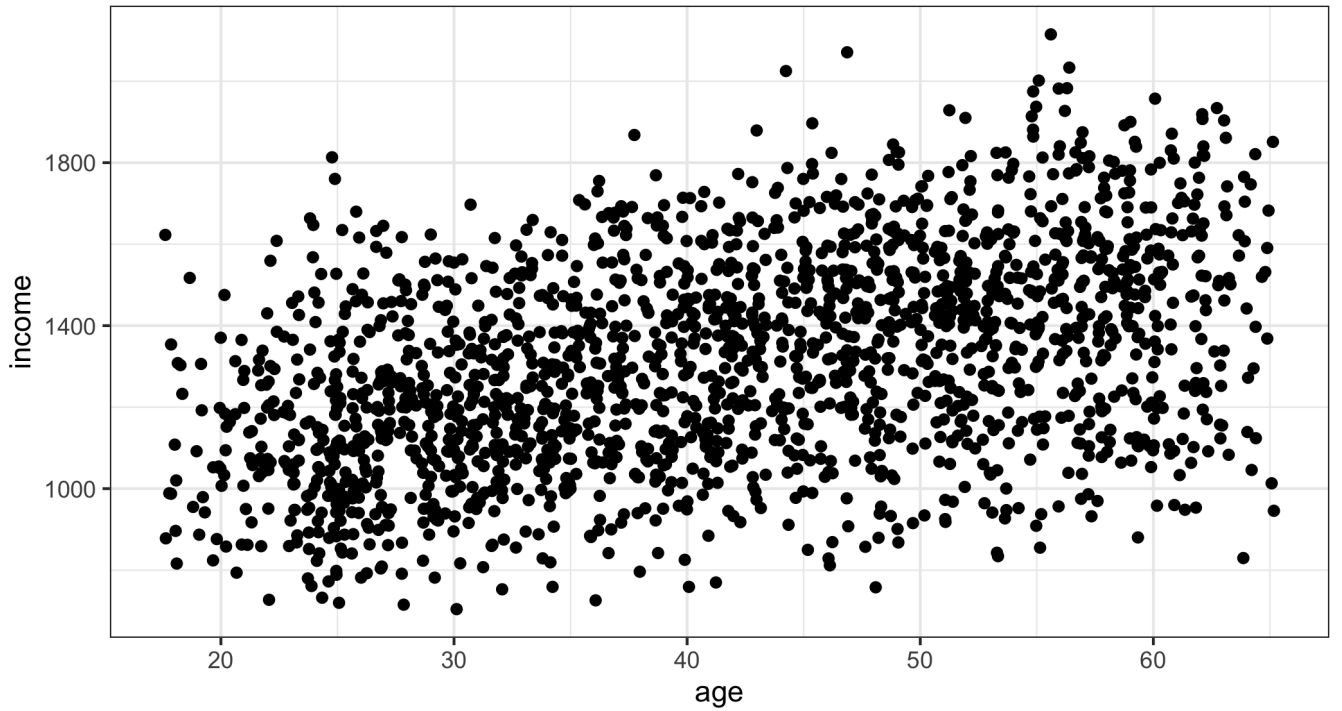
- `geom_point()` with two continuous variables

```
ggplot(incex, aes(x = age, y = income)) + geom_point() + theme_bw()
```



Jittered Scatterplot

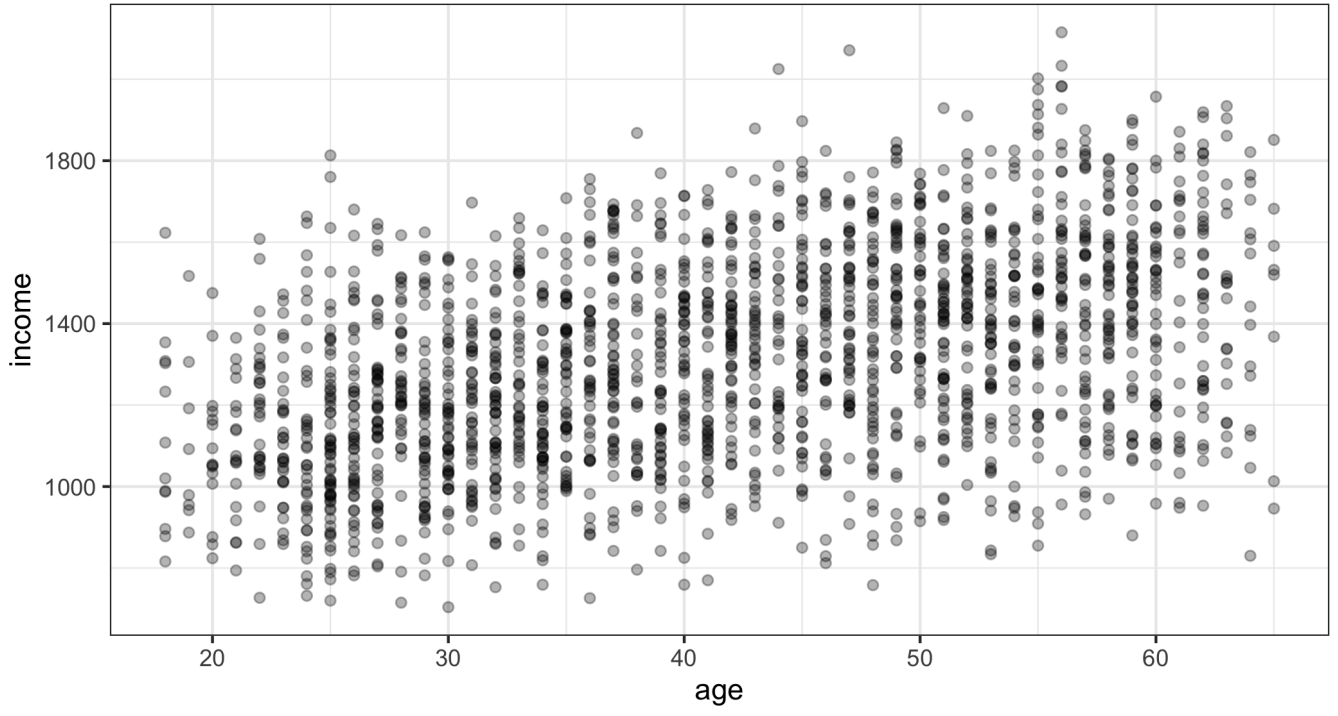
```
ggplot(incex, aes(x = age, y = income)) + geom_jitter() + theme_bw()
```



- “jittering” adds a small random quantity (uniformly distributed) to each observation.

Scatterplot with Different Transparency Levels

```
ggplot(incex, aes(x = age, y = income)) + geom_point(alpha=0.3) + the
```

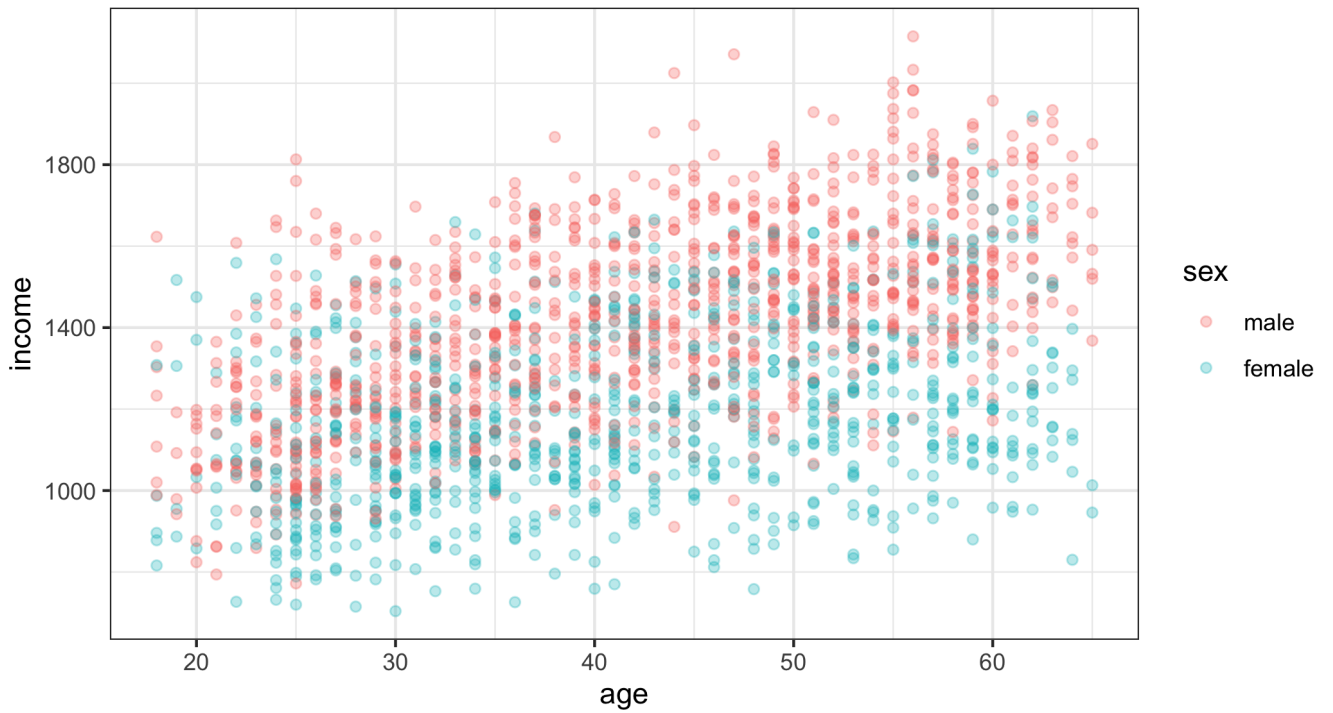


- `alpha` controls the degree of transparency for data points.

Scatterplot with Groups

- Use the argument `col` inside `aes()`.

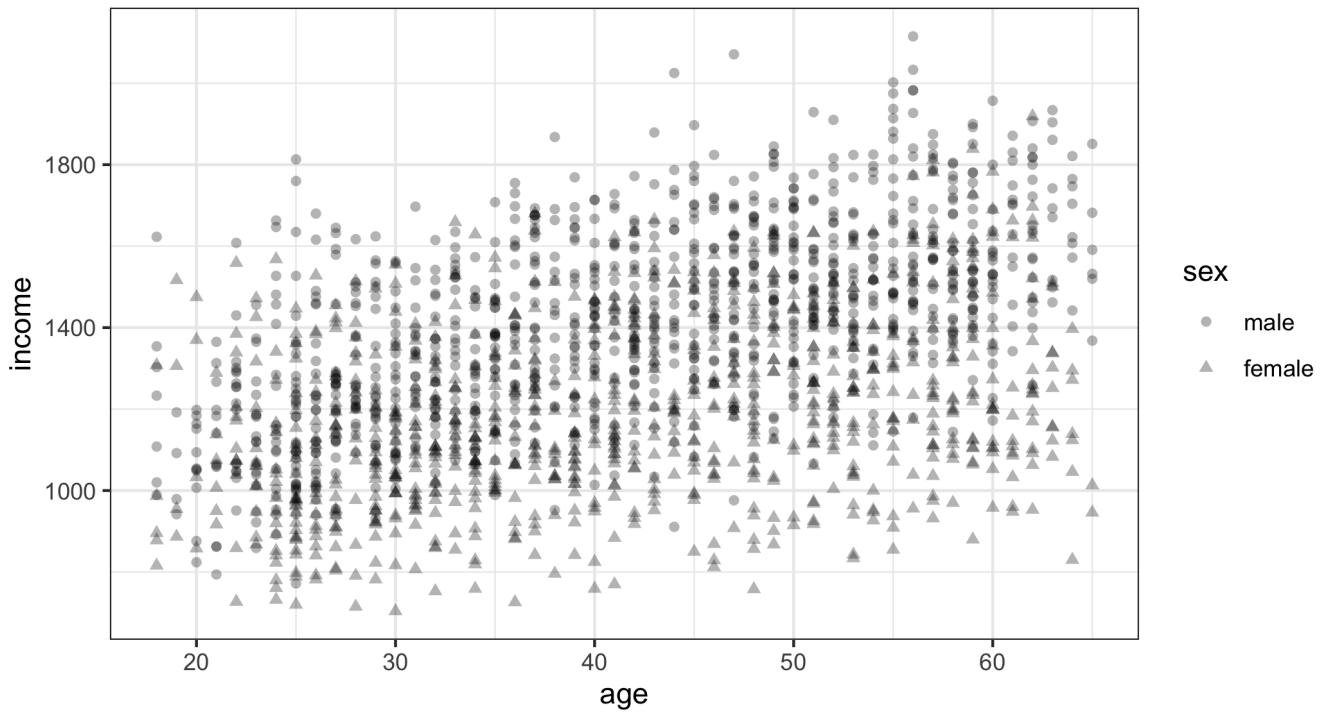
```
ggplot(incex, aes(x = age, y = income, col=sex)) +
  geom_point(alpha=0.3) + theme_bw()
```



Scatterplot with Groups

- Use the argument `shape` inside `aes()`.

```
ggplot(incex, aes(x = age, y = income, shape=sex)) +
  geom_point(alpha=0.3) + theme_bw()
```



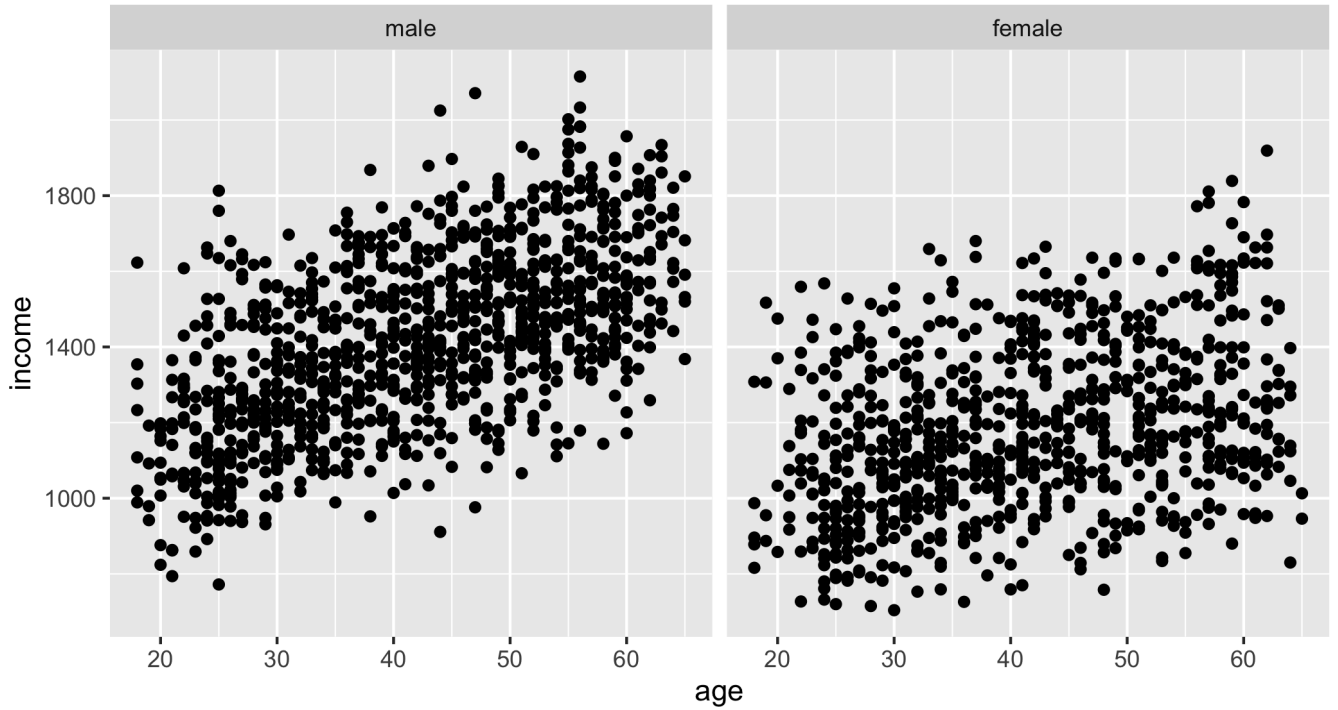
Facets

- Two facet functions for splitting data by categories
- `facet_wrap()` : "wraps" a 1d ribbon of panels into 2d.
- `facet_grid()` : produces a 2d grid of panels defined by variables which form the rows and columns.

Facets: facet_wrap()

- Use `facet_wrap()` with one categorical variable

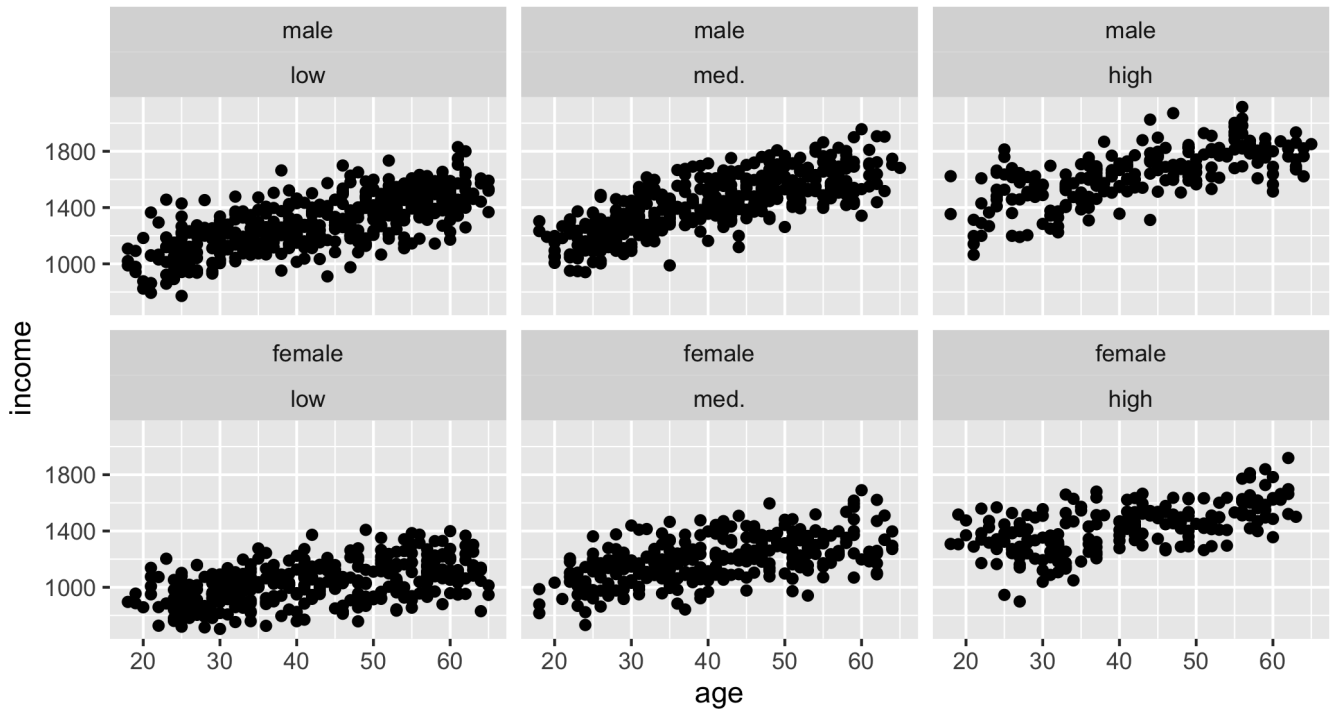
```
ggplot(incex, aes(x = age, y = income)) + geom_point() +
  facet_wrap(~ sex)
```



Facets: facet_wrap()

- Use `facet_wrap()` with two categorical variables: *sex* and *edu*

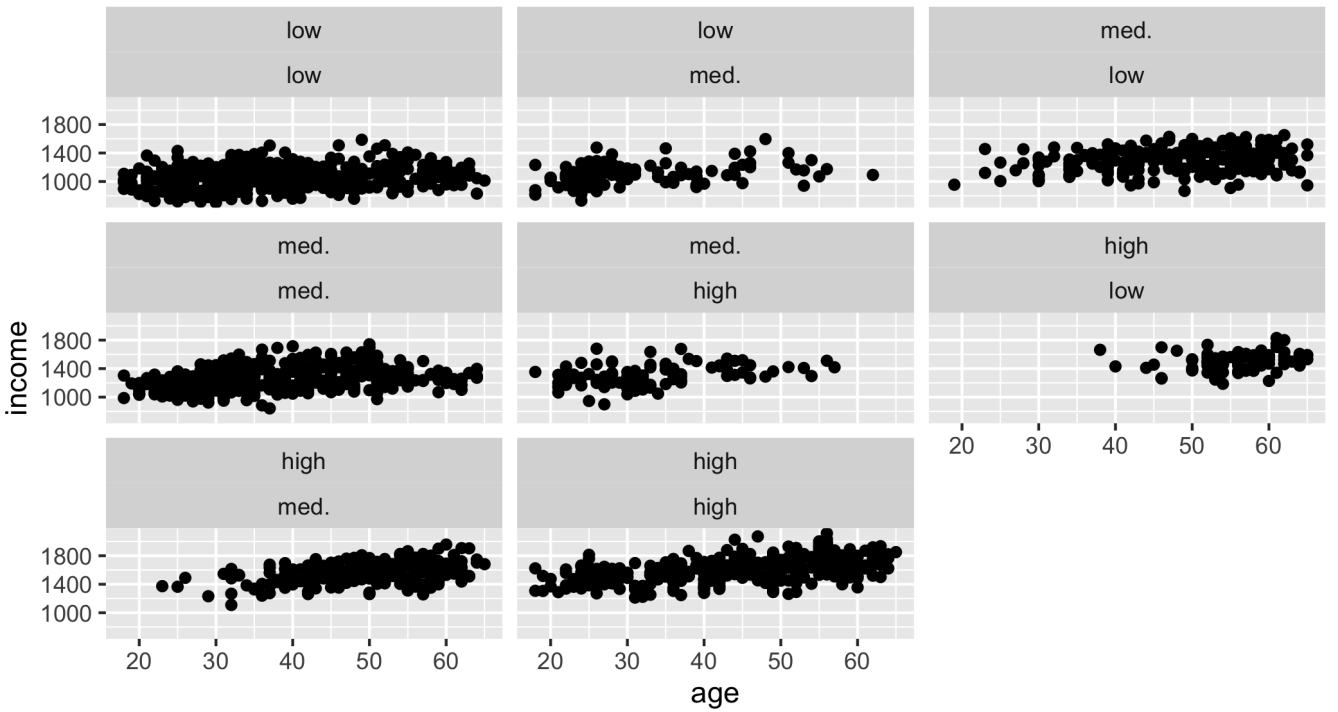
```
ggplot(incex, aes(x = age, y = income)) + geom_point() +
  facet_wrap(sex ~ edu)
```



Facets: facet_wrap()

- Use `facet_wrap()` with two categorical variables: *occ* and *edu*

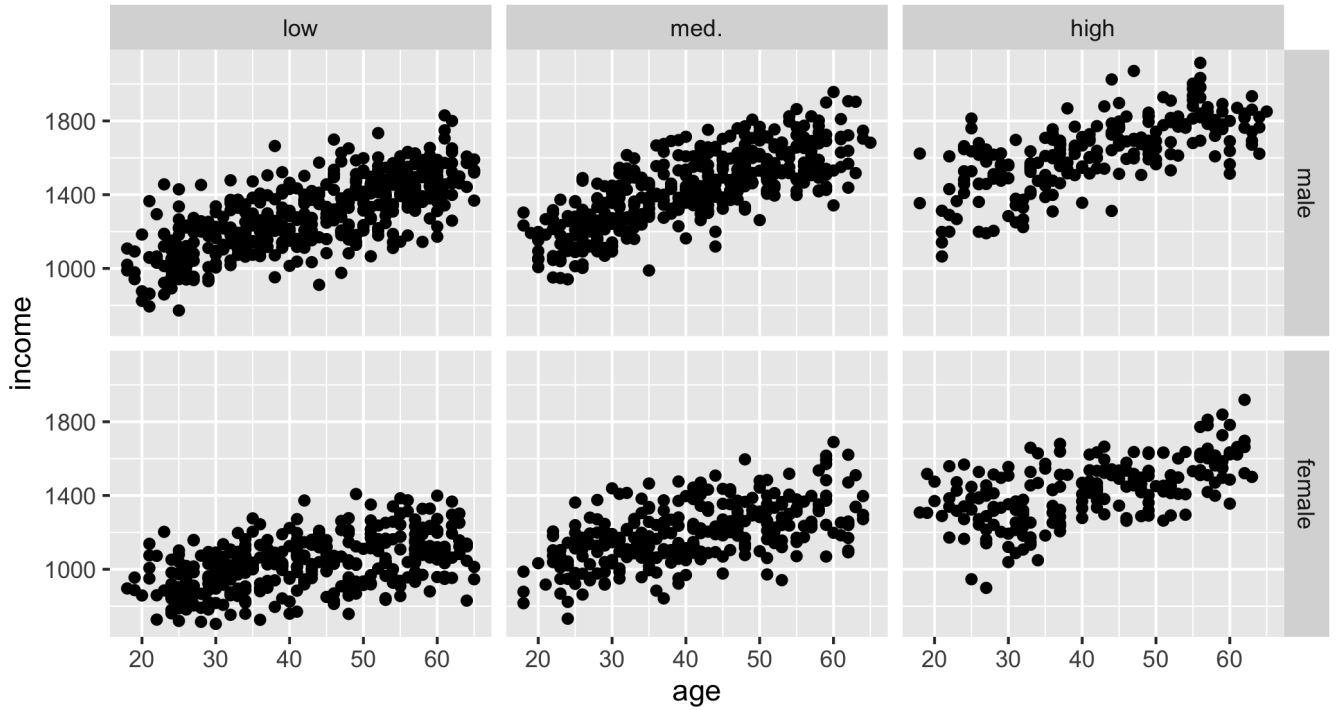
```
ggplot(incex, aes(x = age, y = income)) + geom_point() +
  facet_wrap(occ ~ edu)
```



Facets: facet_grid()

- Use `facet_grid()` with variables *sex* and *edu*.

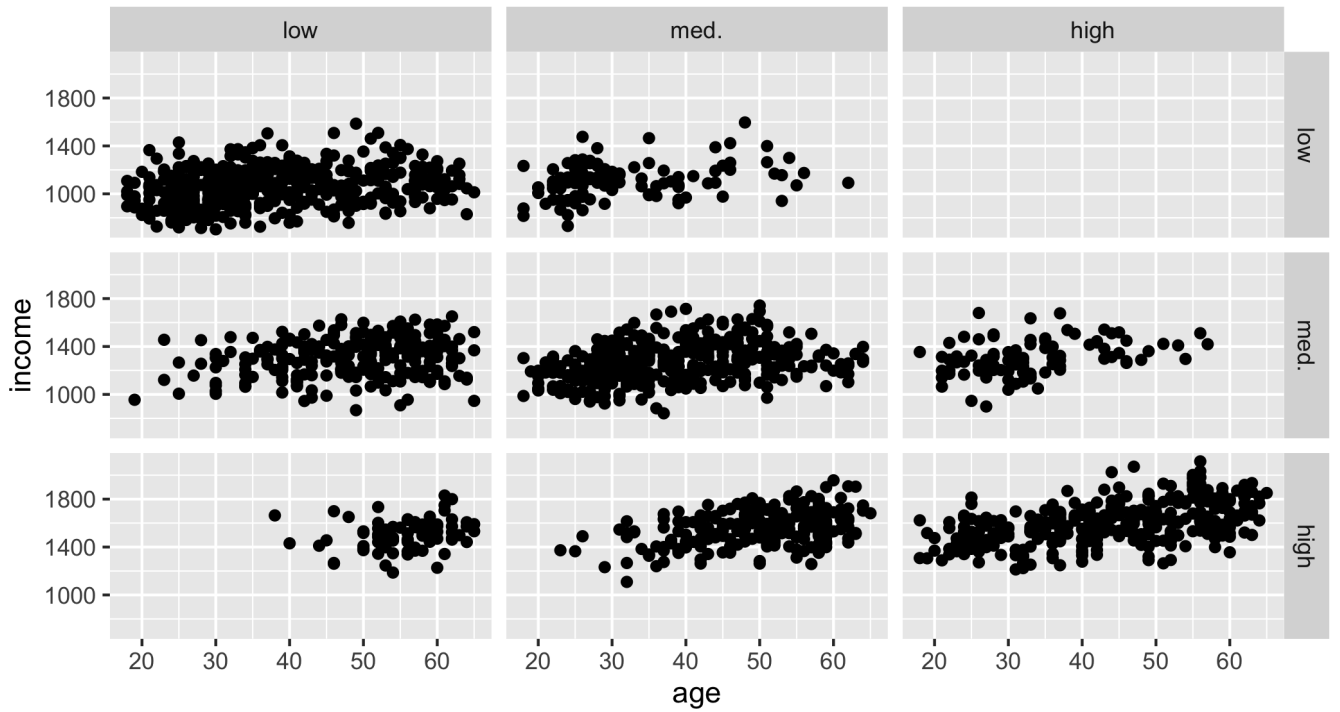
```
ggplot(incex, aes(x = age, y = income)) + geom_point() +
  facet_grid(sex ~ edu)
```



Facets: facet_grid()

- Use `facet_grid()` with variables *occ* and *edu*.

```
ggplot(incex, aes(x = age, y = income)) + geom_point() +
  facet_grid(occ ~ edu)
```

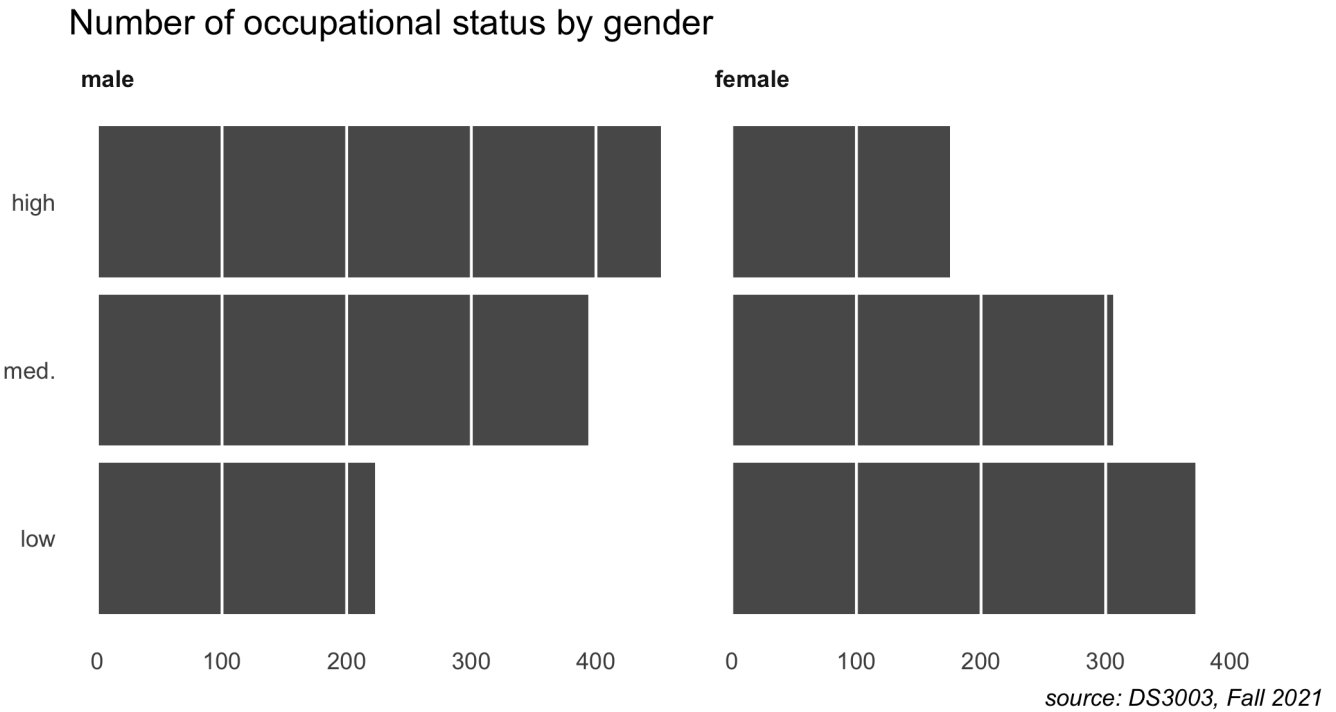


Themes

- Try different arguments for `theme`!

```
ggplot(incex) + geom_bar(aes(y = occ)) + facet_wrap(~ sex) +
    labs(title = "Number of occupational status by gender",
         caption = "source: DS3003, Fall 2021",
         x = NULL,
         y = NULL) +
    theme_minimal() +
    theme(
      strip.text = element_text(face = 'bold', hjust = 0),
      plot.caption = element_text(face = 'italic'),
      panel.grid.major = element_line('white', size = 0.5),
      panel.grid.minor = element_blank(),
      panel.grid.major.y = element_blank(),
      panel.ontop = TRUE
    )
```

Themes


Number of occupational status by gender
male
female
high
med.
low
0
100
200
300
400
0
100
200
300
400
source: DS3003, Fall 2021

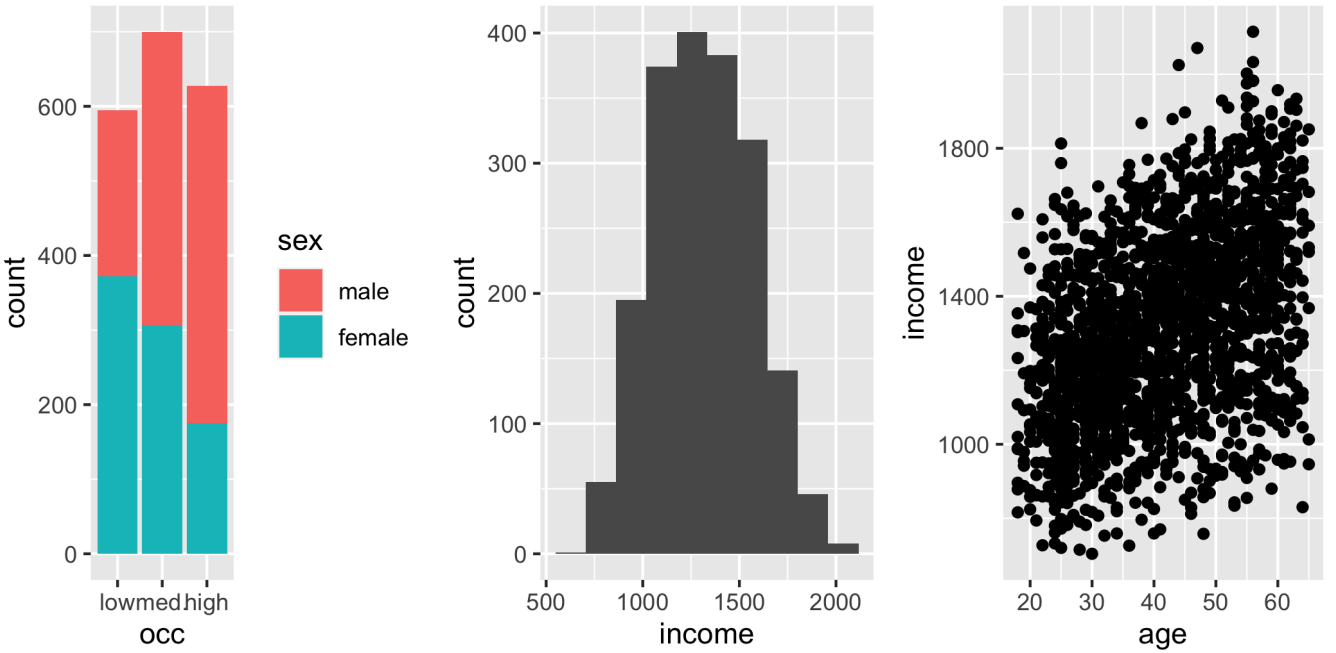
Beyond ggplot2

- ggplot2 is huge! About 50 geoms, 25 stats, 60 scales.
- Many extensions are very niche specific and developed by experts in the field.
- 100 registered extensions available to explore (https://exts.ggplot2.tidyverse.org)
- e.g., for plot compositions, you might want to use `gridExtra::grid.arrange()`, `ggpubr::ggarrange()`, `cowplot::plot_grid()`, or `patchwork`.

Example: package gridExtra

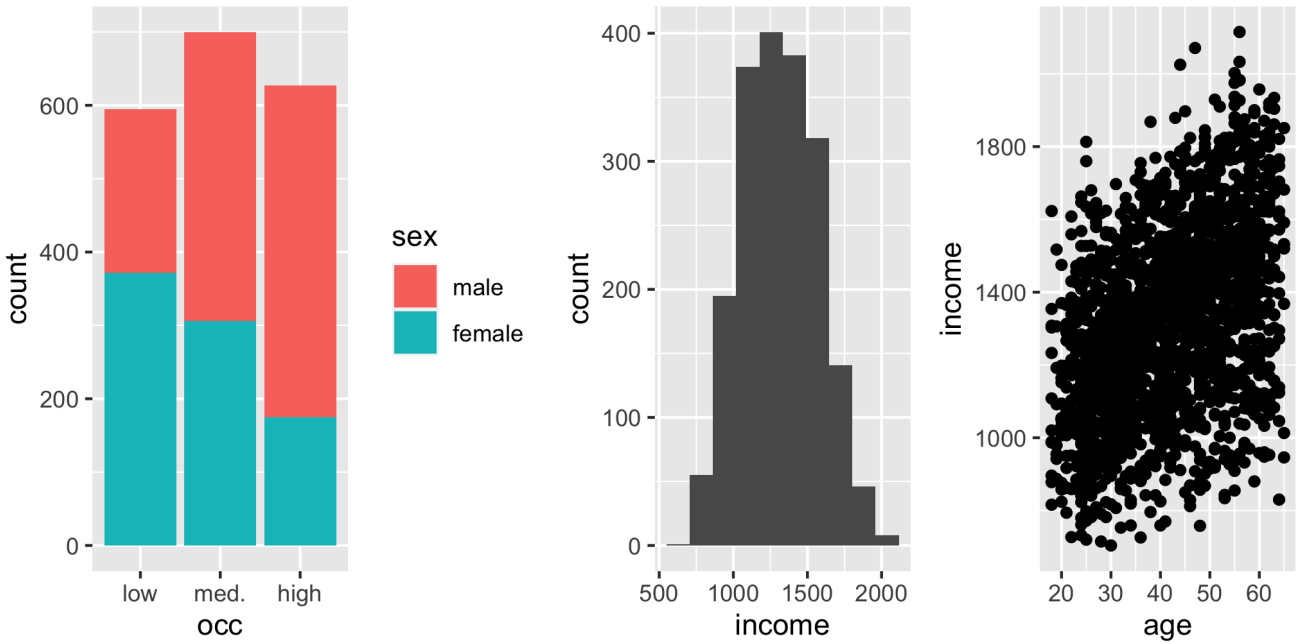
```
library(gridExtra)
p1 <- ggplot(incex) + geom_bar(aes(x = occ, fill=sex)) # barplot
p2 <- ggplot(incex) + geom_histogram(aes(x=income), bins=10) # histog
p3 <- ggplot(incex) + geom_point(aes(x = age, y=income)) # scatterplo

grid.arrange(p1, p2, p3, nrow=1)
```



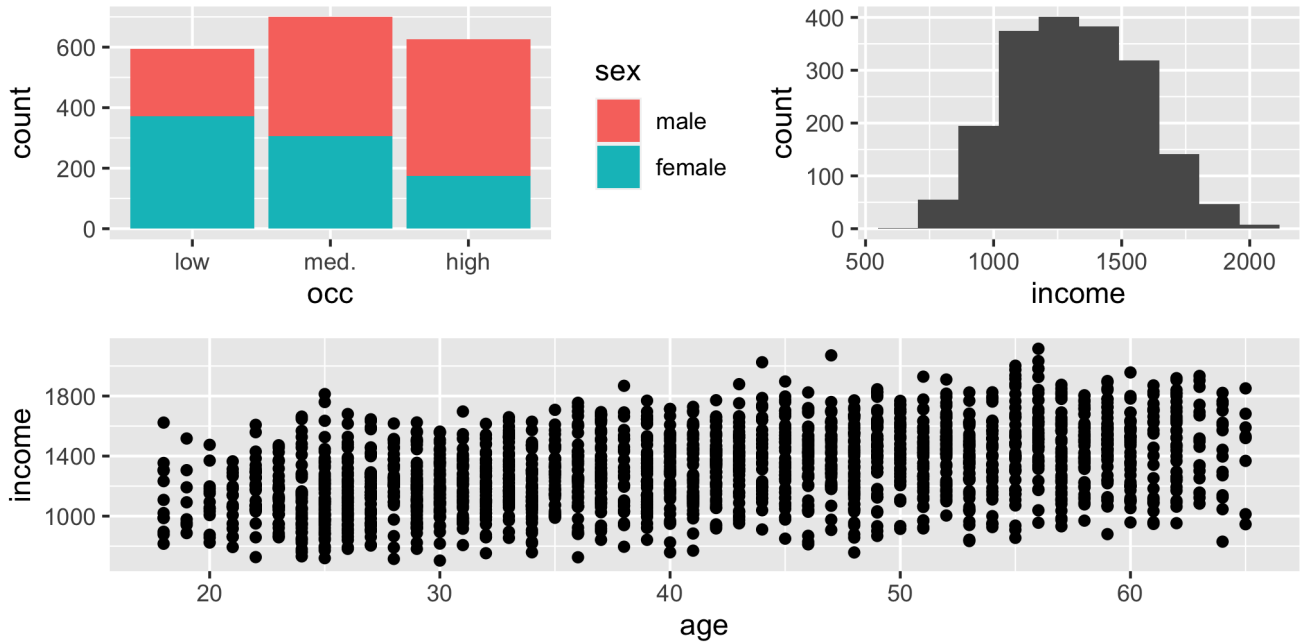
Example: package patchwork

```
library(patchwork)
p1 + p2 + p3
```



Example: package patchwork

```
library(patchwork)
( p1 + p2 ) / p3
```



Example: package GGally

```
library(GGally)
ggpairs(incex[, c('income', 'oexp', 'age', 'edu')])
```