Visualization and Exploration

January 25, 2016

Visualization

Reasons to use visualization:

- to find problems with the data
- to explore dependencies and features
- to present results

General guidelines:

- Display as much information as possible with least amount of effort required from the viewer to get it.
- Clarity is paramount - make the data stand out
  - avoid overusing colours, shapes, patterns
  - avoid distracting elements that don't add value, e.g. grid lines, background colours
  - use the right aspect ratio
- Visualization is an iterative process

Visualization in R

Standard graphing capabilities in R are the `graphics` package.

Package `lattice` improves by adding easy display of multivariate and conditional relationships. Implementation of the *trellis* project:

`http://ect.bell-labs.com/sl/project/trellis/`

See chapters 3 and 4 in "Using R for Data Analysis and Graphics" for introduction and examples:

`http://cran.r-project.org/doc/contrib/usingR.pdf`

Also try:

```
demo(graphics)
demo(lattice)
```

Visualization in R

Package `ggplot2` is the new kid on the block. Implements the *The Grammar of Graphics* by Leland Wilkinson:

`https://www.springer.com/statistics/computational+statistics/book/978-0-387-24544-7`

Documentation at `http://docs.ggplot2.org/current/`

- In `ggplot2` graphs are defined on data frames.
- Graphs are produced by *adding* layers and transformations.
- Data are displayed using *aesthetics*, such as position, colour, size, shape

Package `ggplot2`

Some graph elements in `ggplot2`:

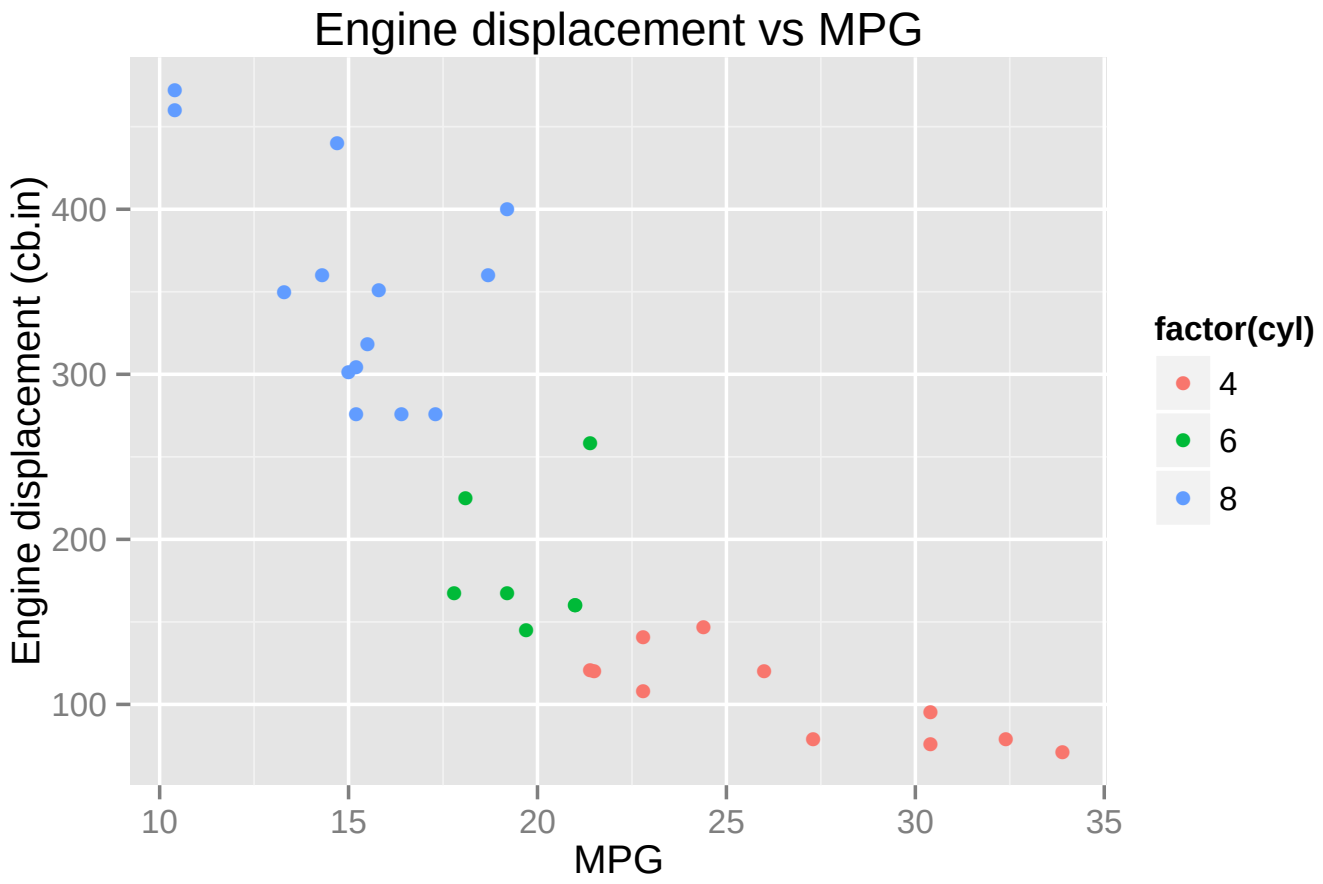
- `geom`: geometric objects define the type of plot
- `stat`: statistical transformations
- `facet`: displays subsets of the data in different panels allowing for visualization of conditional relationships.

Use `ggplot` function to create graph object and add layers with the `+` operator.

Use `qplot` function for a simplified interface to `ggplot2`.

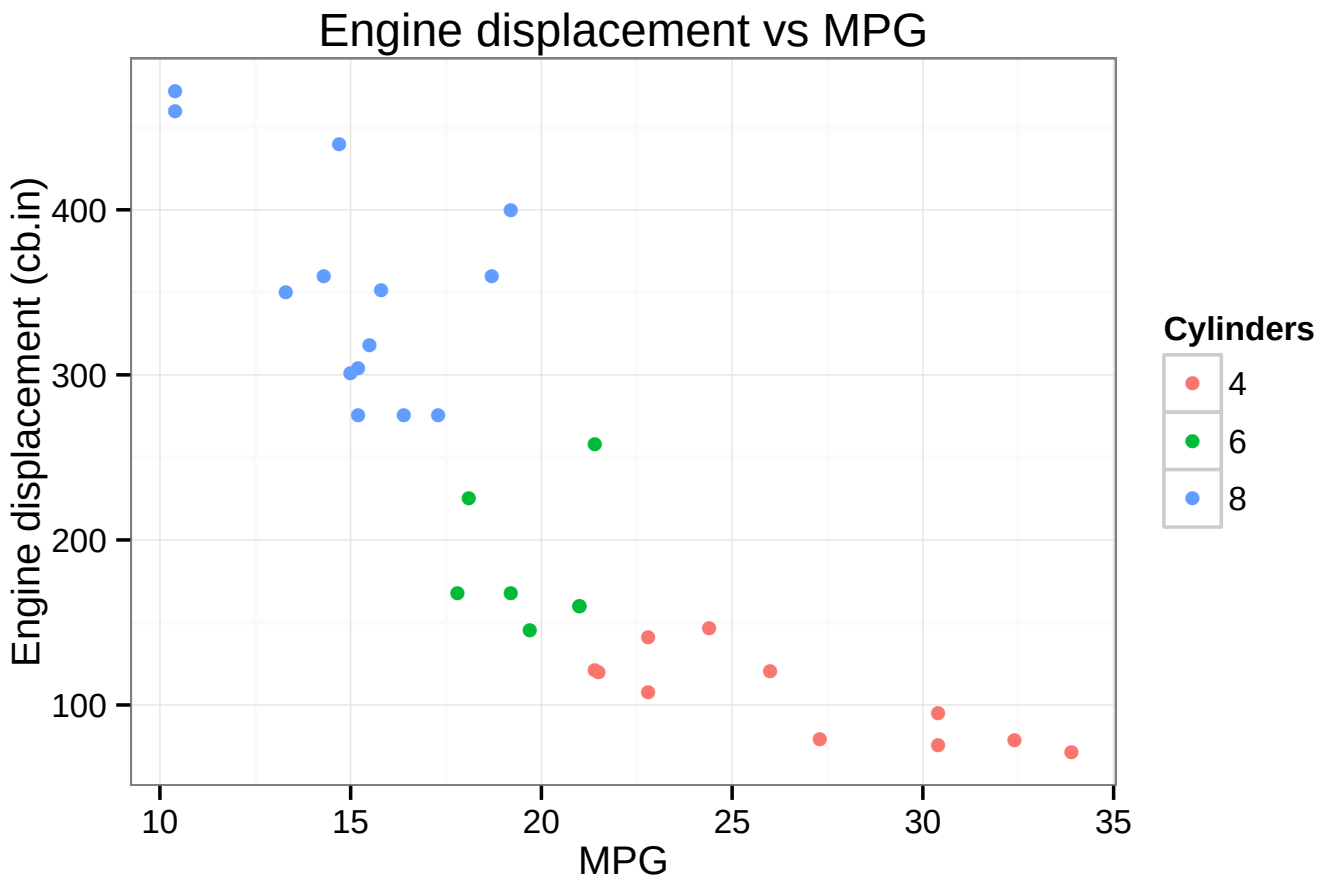
Example

```
qplot(mpg, disp, data=mtcars, colour=factor(cyl),
    main="Engine displacement vs MPG", xlab="MPG",
    ylab="Engine displacement (cb.in)")
```



Example

```
qplot(mpg, disp, data=mtcars, colour=factor(cyl),
    main="Engine displacement vs MPG", xlab="MPG",
    ylab="Engine displacement (cb.in)") +
  theme_bw() + labs(colour="Cylinders")
```



Dataset for examples

A copy of the file is available on the course webpage.

```
custdata <- read.table("custdata.tsv", header=T, sep="\t")
```

The business objective is to predict whether your customer has health insurance. This synthetic dataset contains customers information for ones whose health insurance status is known.

Spot problems

Missing values

```
> dim(custdata)
[1] 1000   11
> mv <- colSums(is.na(custdata))
> cbind(mv)    % cbind to display as column
                mv
custid           0
sex              0
is.employed    328
income           0
marital.stat     0
health.ins       0
housing.type    56
recent.move     56
num.vehicles    56
age              0
state.of.res     0
```

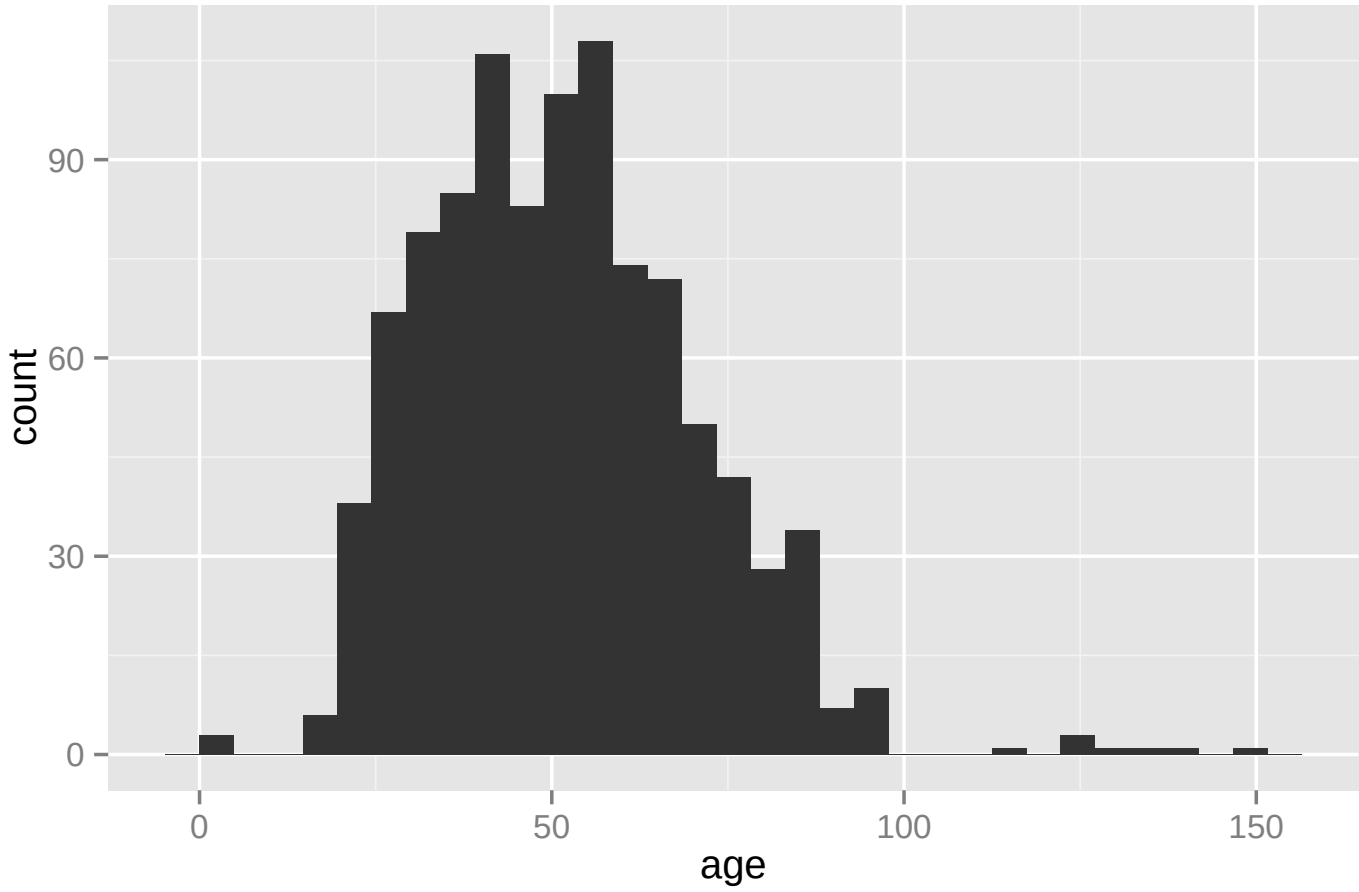
Spot problems

Values out of range

```
> summary(custdata$income)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -8700   14600   35000   53500   67000  615000
>
> summary(custdata$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    38.0    50.0    51.7    64.0   146.7
```

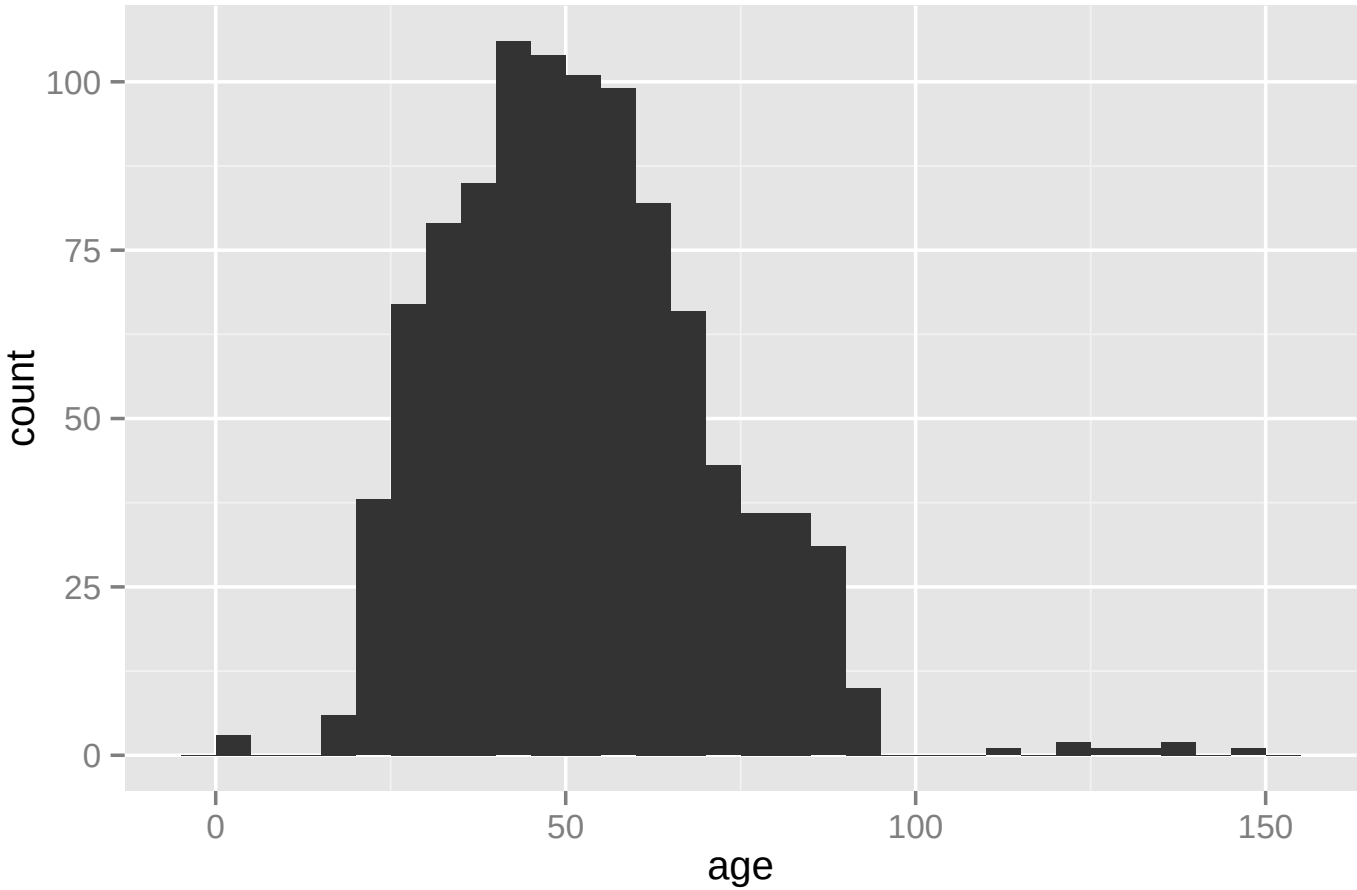
Values out of range

```
> qplot(age, data=custdata)
stat_bin: binwidth defaulted to range/30. Use
'binwidth = x' to adjust this.
```



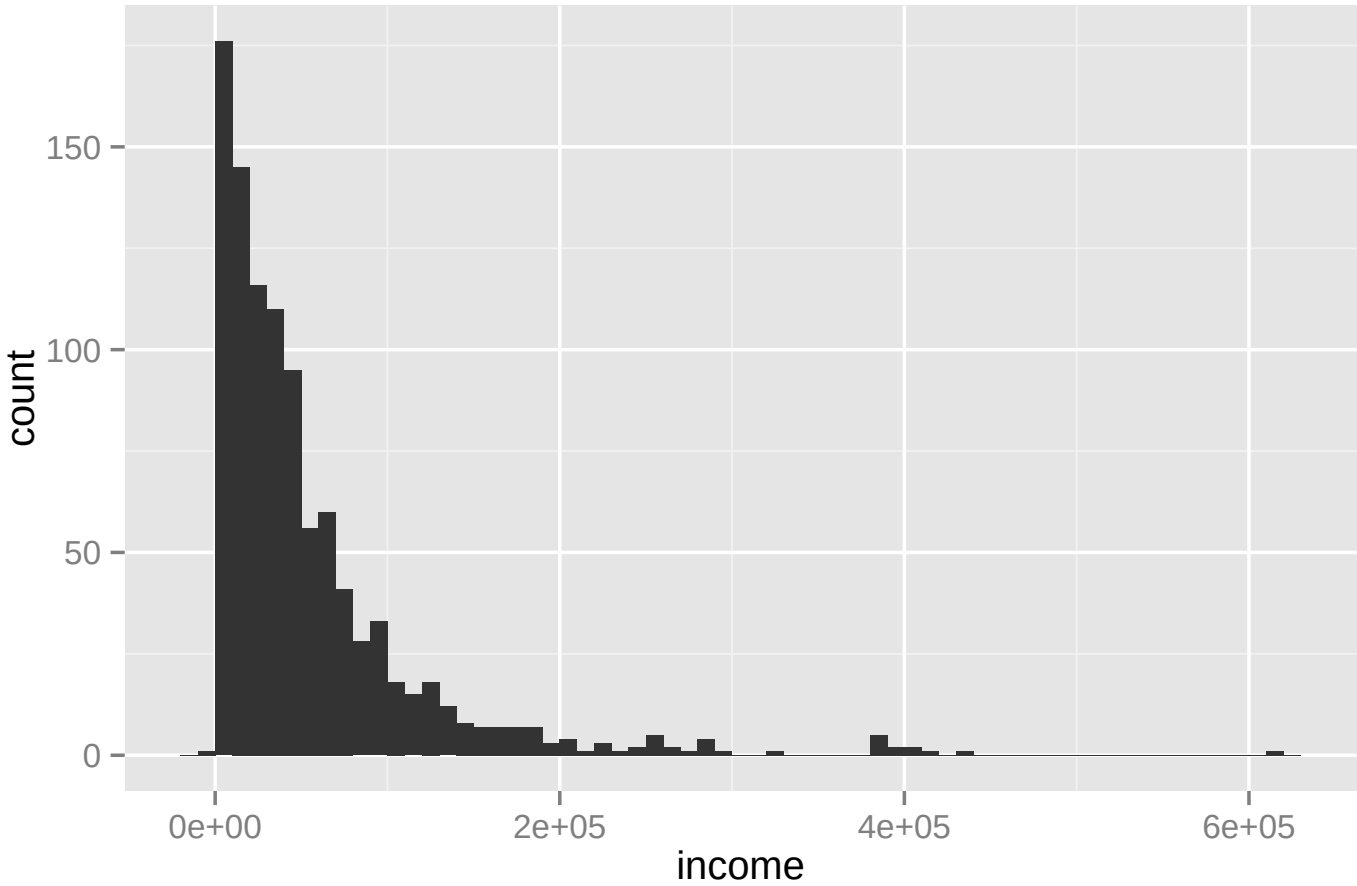
Values out of range

```
qplot(age, data=custdata, binwidth=5)
```



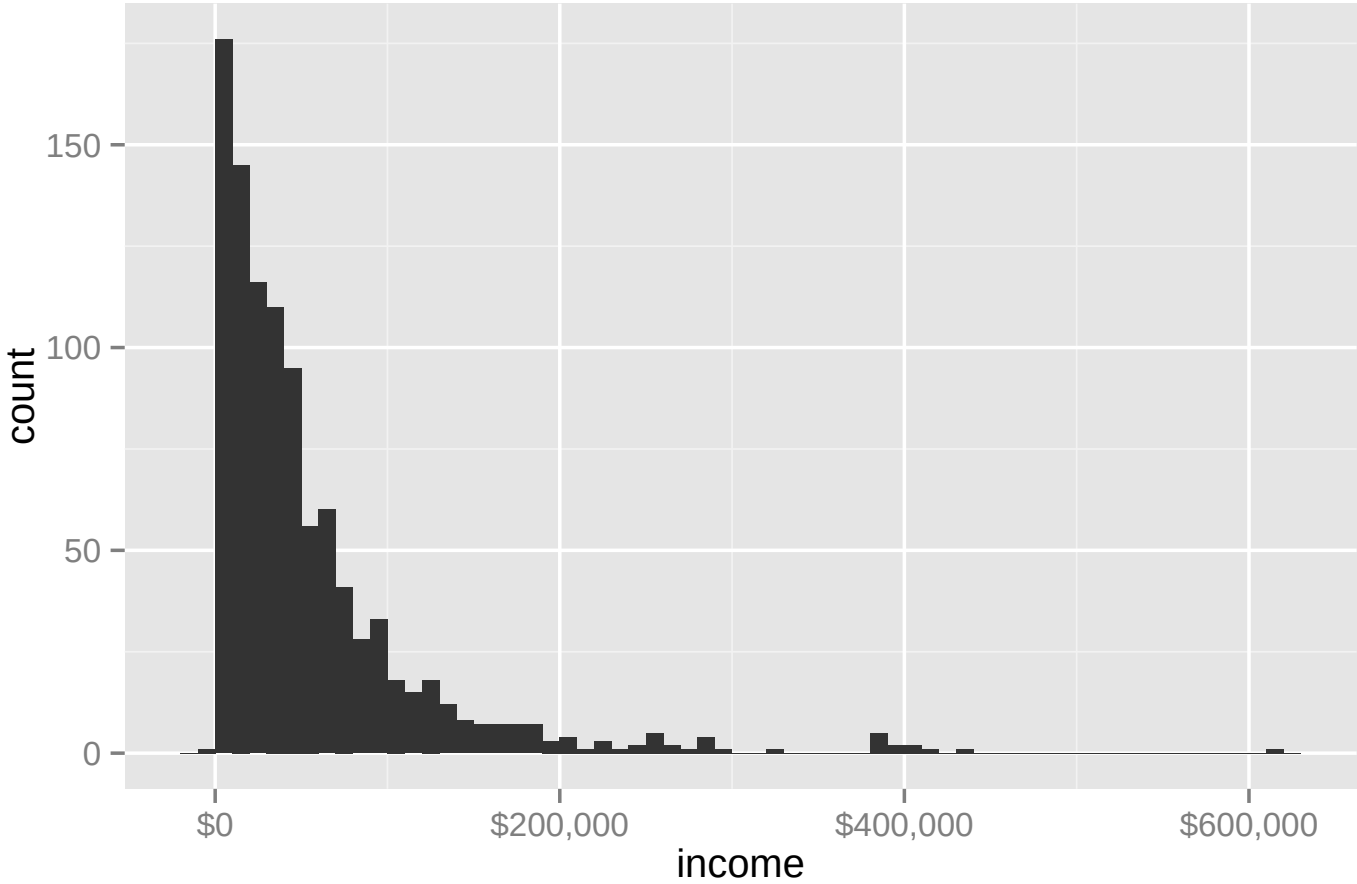
Values out of range

```
qplot(income, data=custdata, binwidth=10000)
```



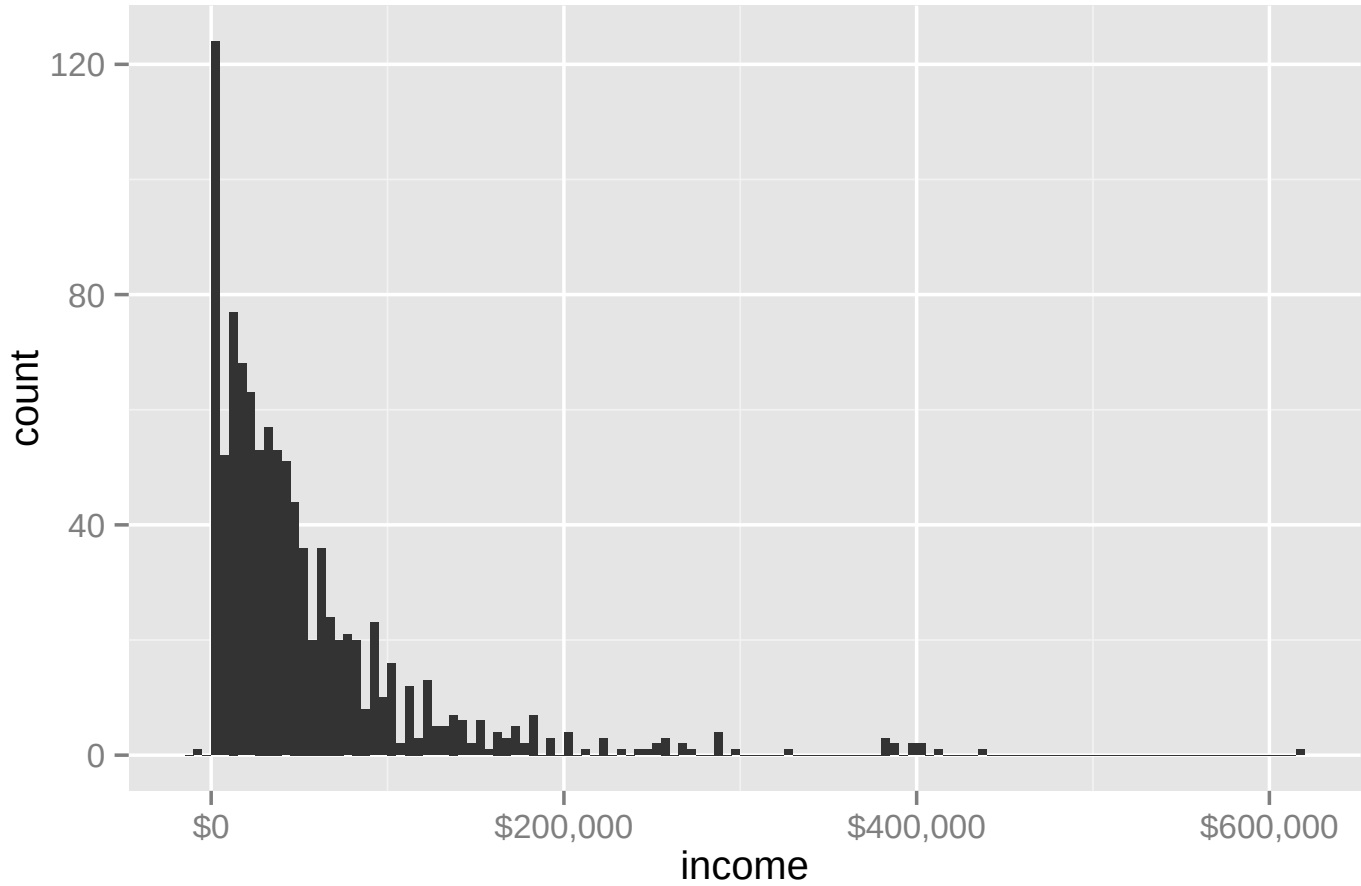
Values out of range

```
library(scales)
qplot(income, data=custdata, binwidth=10000) +
    scale_x_continuous(labels=dollar)
```



Values out of range

```
qplot(income, data=custdata, binwidth=5000) +
    scale_x_continuous(labels=dollar)
```



- `qplot` selects automatically the type of graph from the number and type of arguments
- for a single numerical variable the default is histogram
- the same plot can be done using the following commands

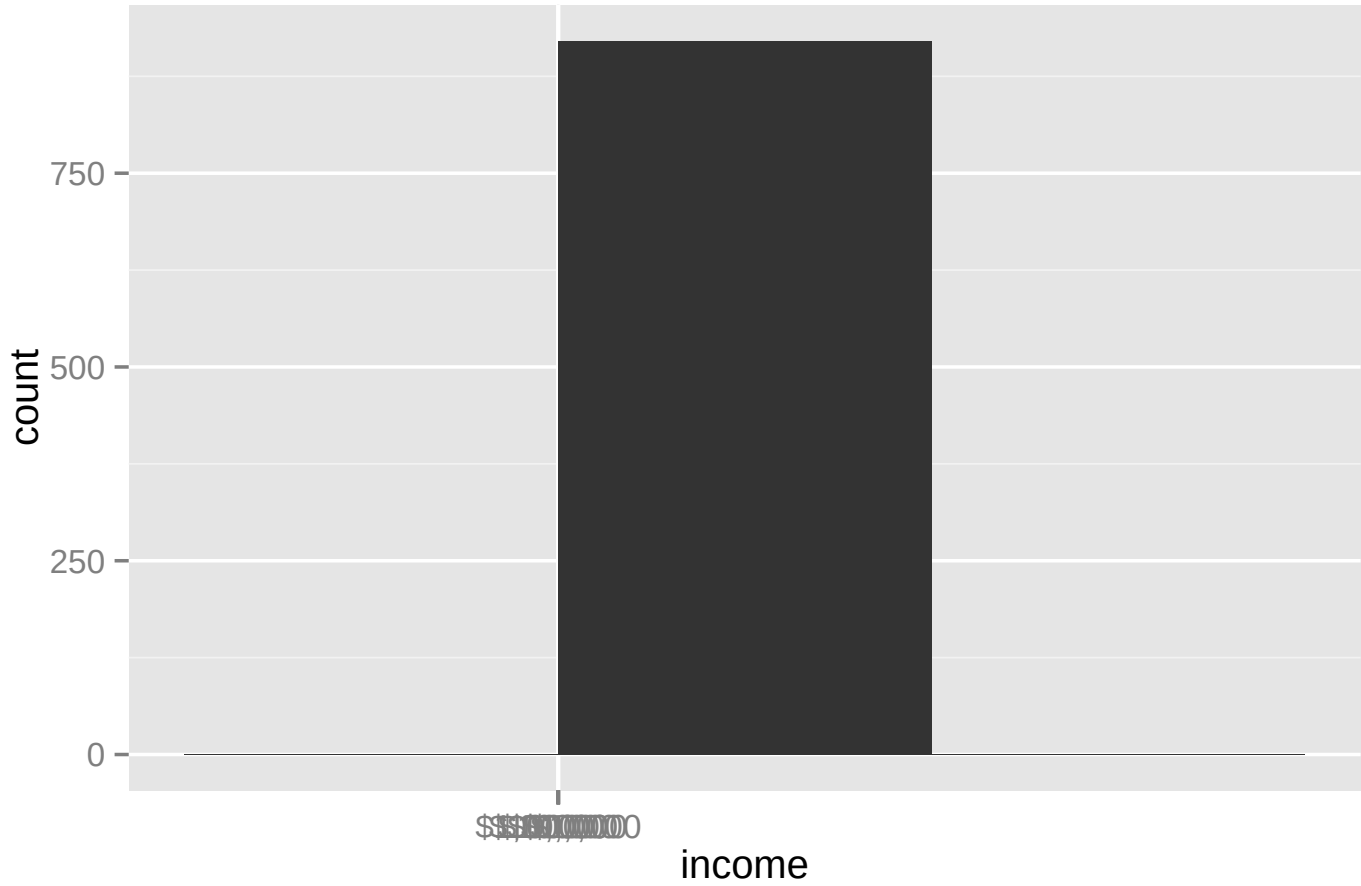
```
ggplot(custdata) +
    geom_histogram(aes(x=income), binwidth=5000) +
    scale_x_continuous(labels=dollar)
```

Logarithmic scale

- Use logarithmic scale for variables where percent change is more important than change in value.
- Use logarithmic scale when data spans a wide range, e.g. multiple orders of magnitude

Logarithmic histogram

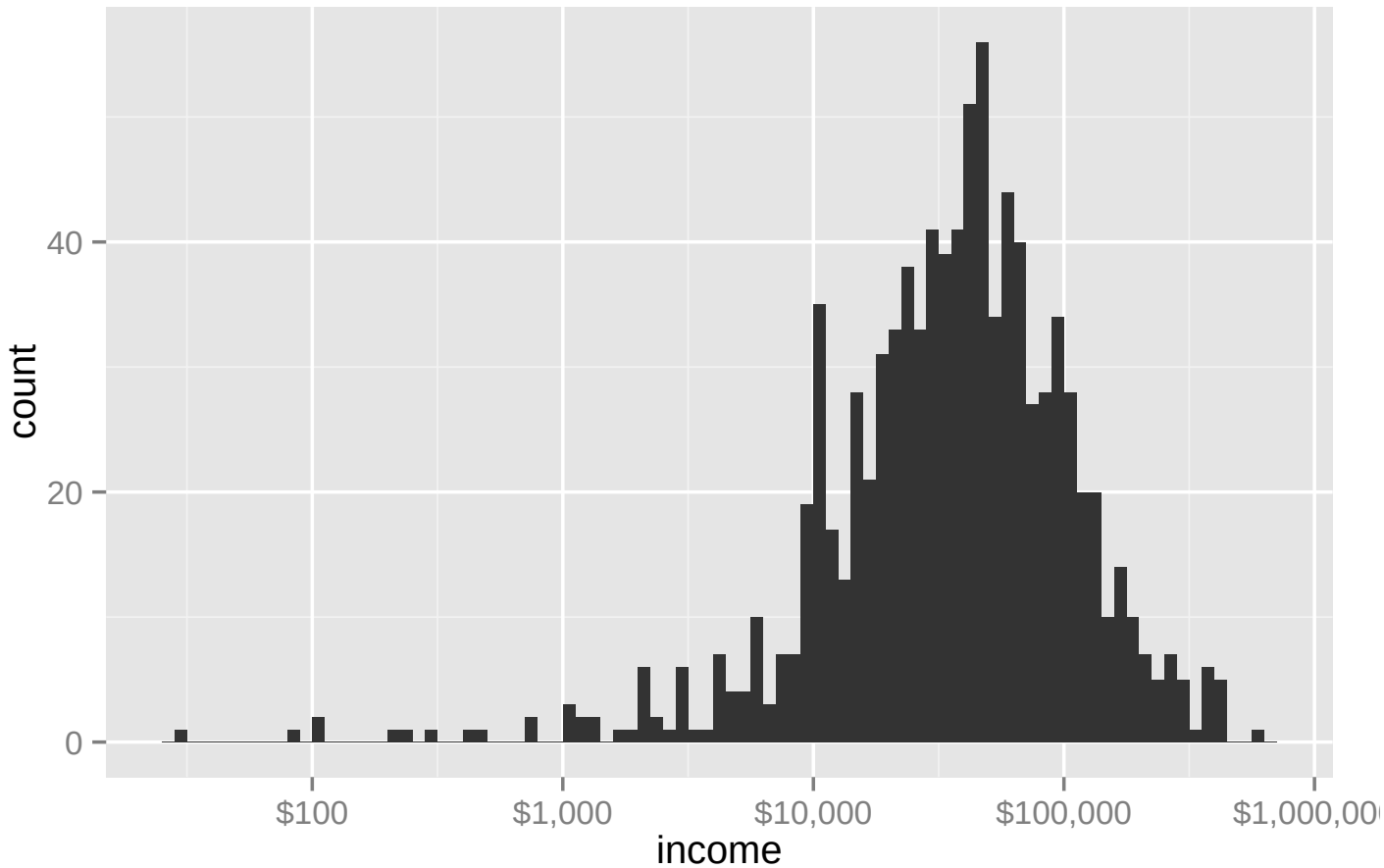
```
custdata2 <- subset(custdata, income > 0)
qplot(income, data=custdata2, binwidth=5000) +
    scale_x_log10(breaks=10^(1:6), labels=dollar)
```



Logarithmic histogram

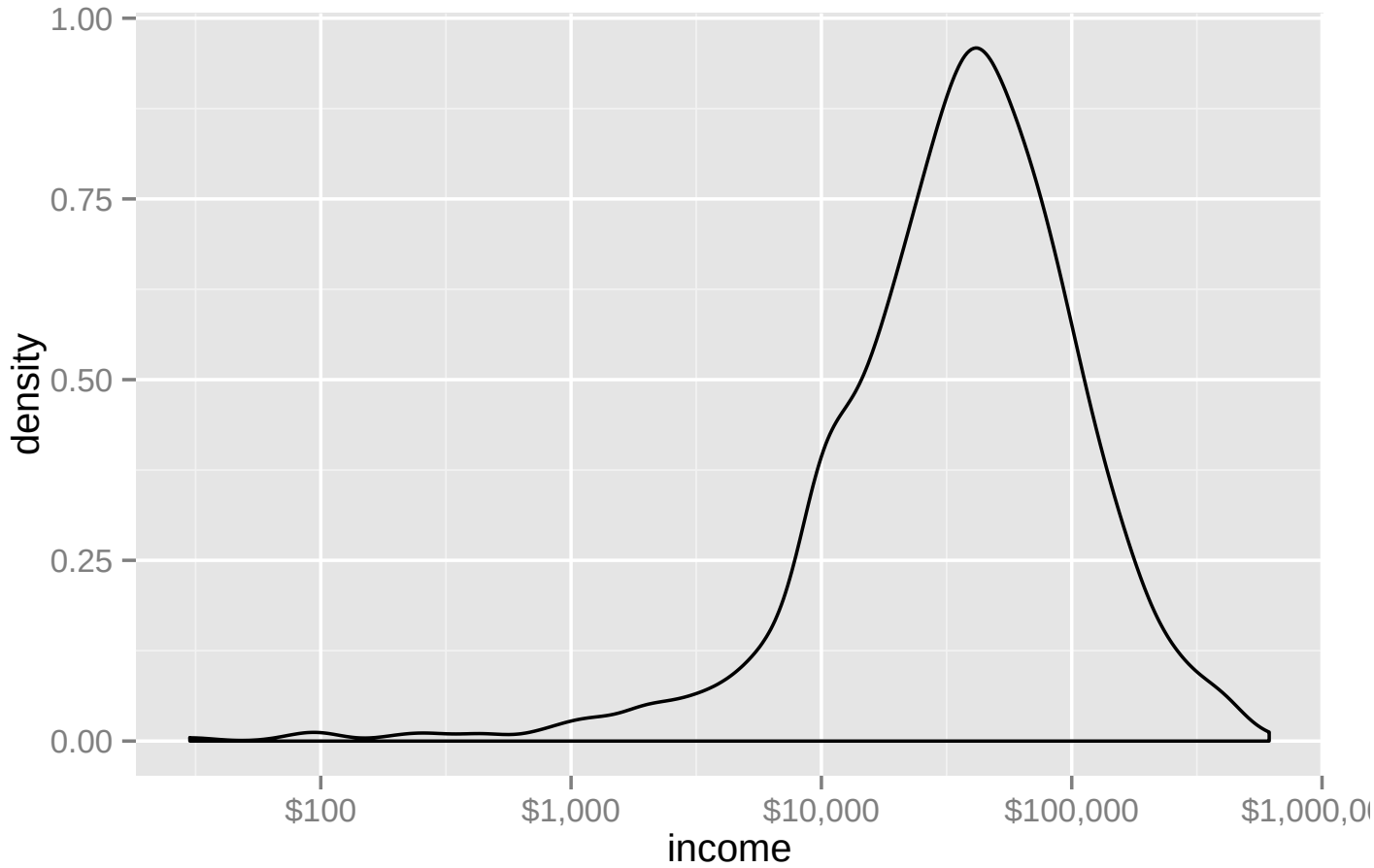
binwidth should be in percent change, not dollar amount

```
qplot(income, data=custdata2, binwidth=0.05) +
    scale_x_log10(breaks=10^(1:6), labels=dollar)
```



Density plots

```
qplot(income, data=custdata2, geom="density") +
    scale_x_log10(breaks=10^(1:6), labels=dollar)
```



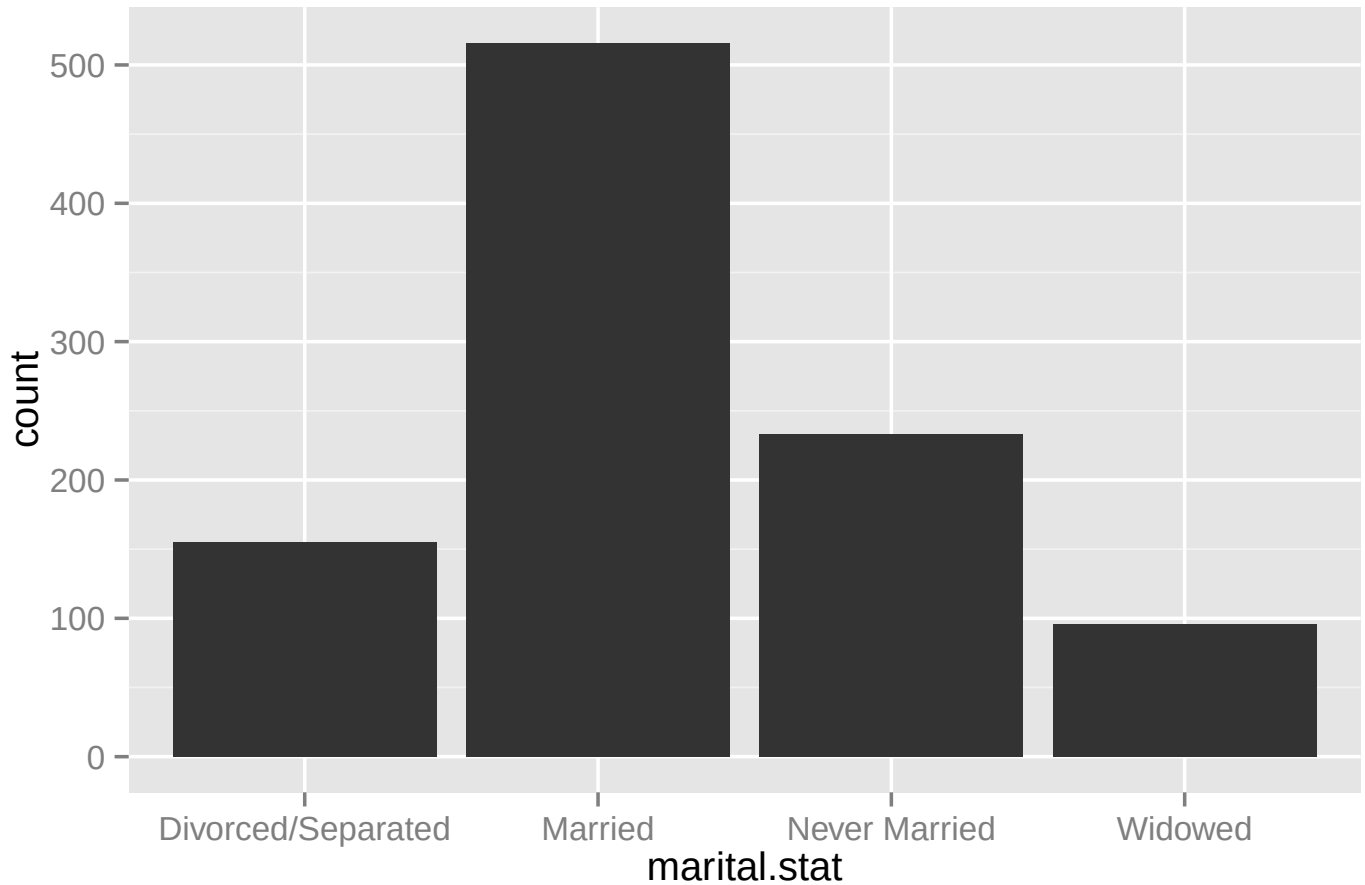
Histogram vs density

- Both apply to continuous variables.
- Both give an idea of the underlying probability distribution.
- Two histograms of the same data may look very different with different bin widths and choosing the best bin width is important.
- A density plot is a *"continuous histogram"*. It plots an estimated probability distribution function.

Bar charts

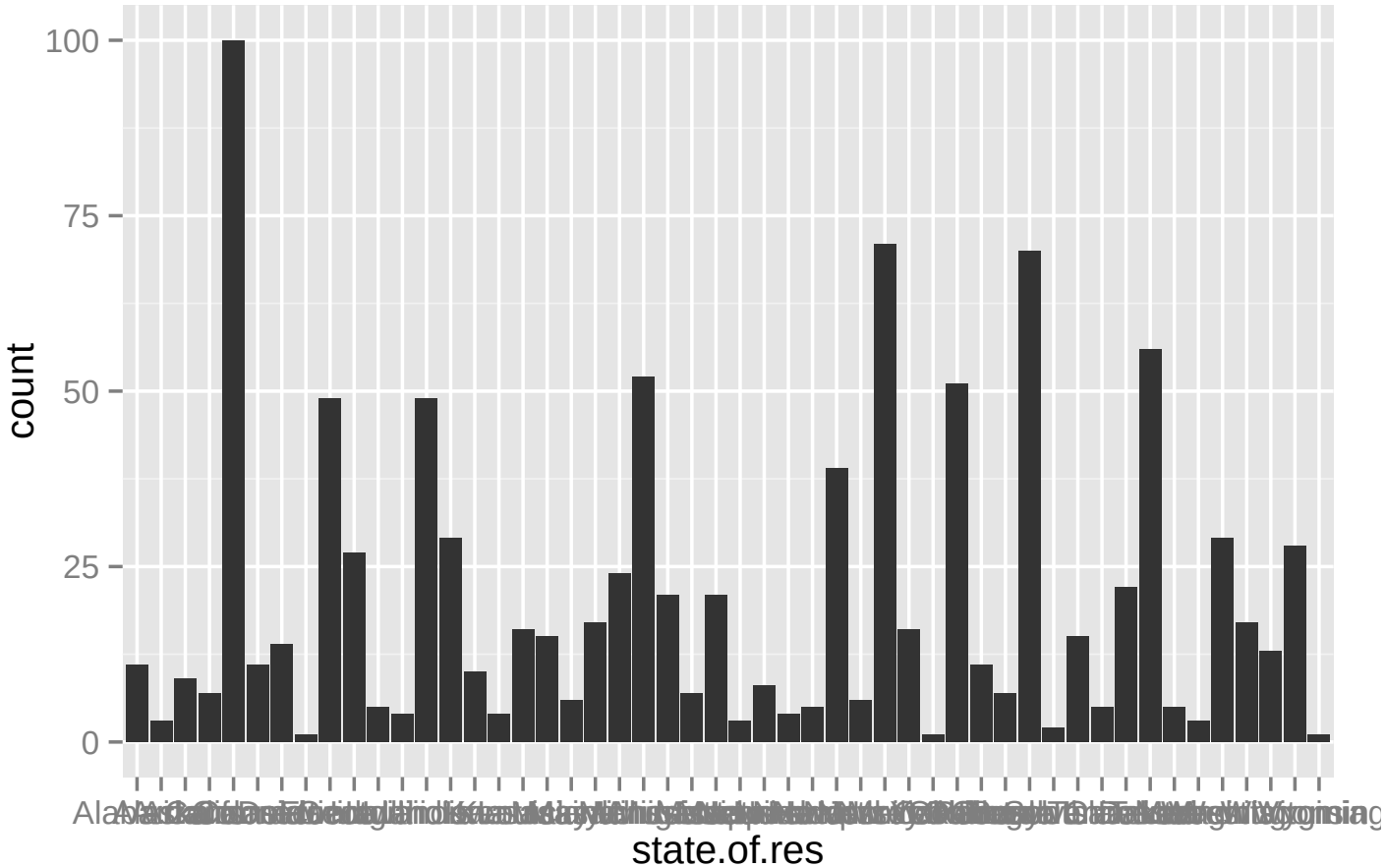
A bar chart is a histogram for categorical variable. It is the default geometry in `qplot` for factor and logical variables

```
qplot(marital.stat, data=custdata)
```



Bar charts

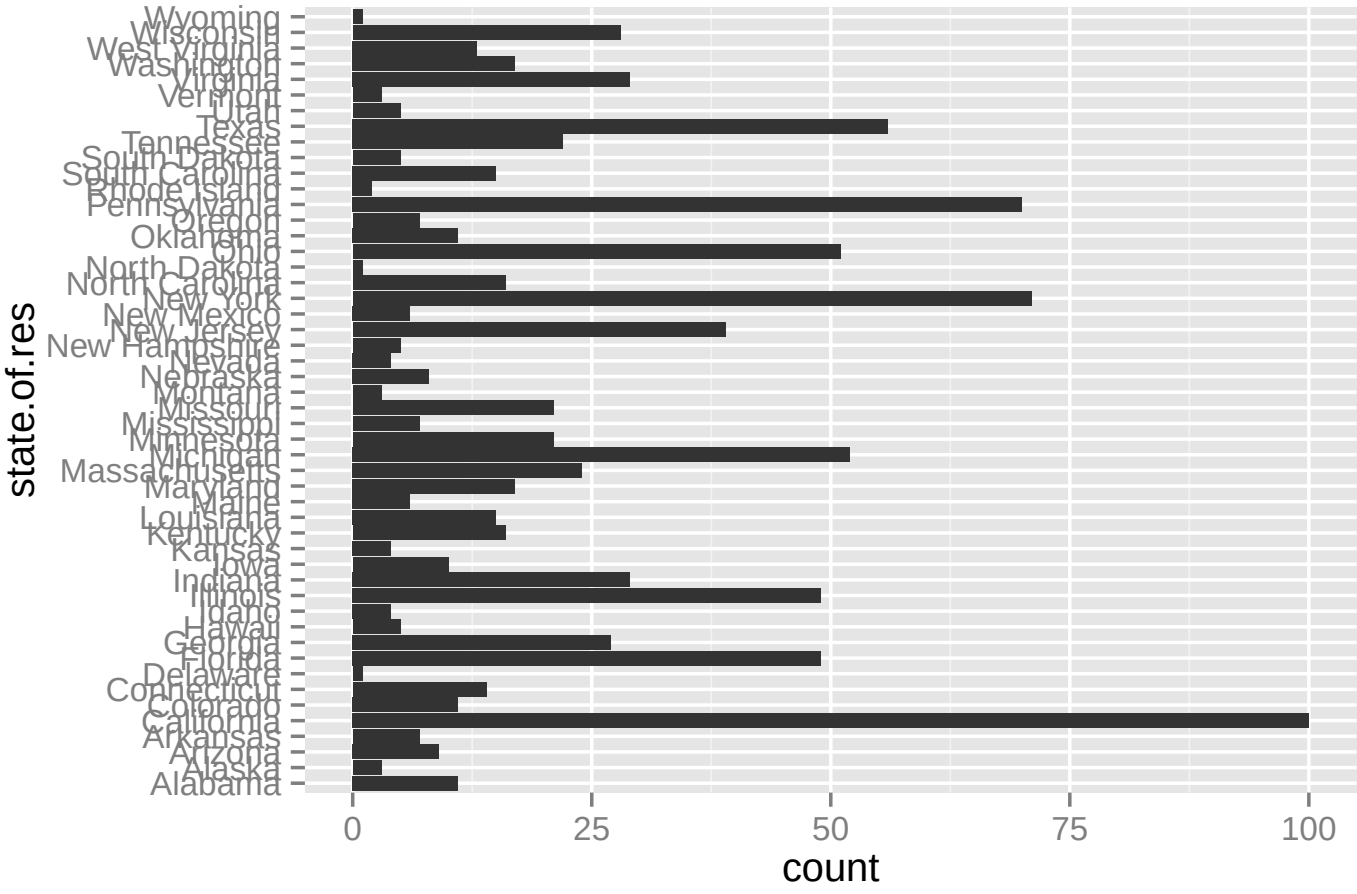
```
qplot(state.of.res, data=custdata)
```



What a mess!

Bar charts

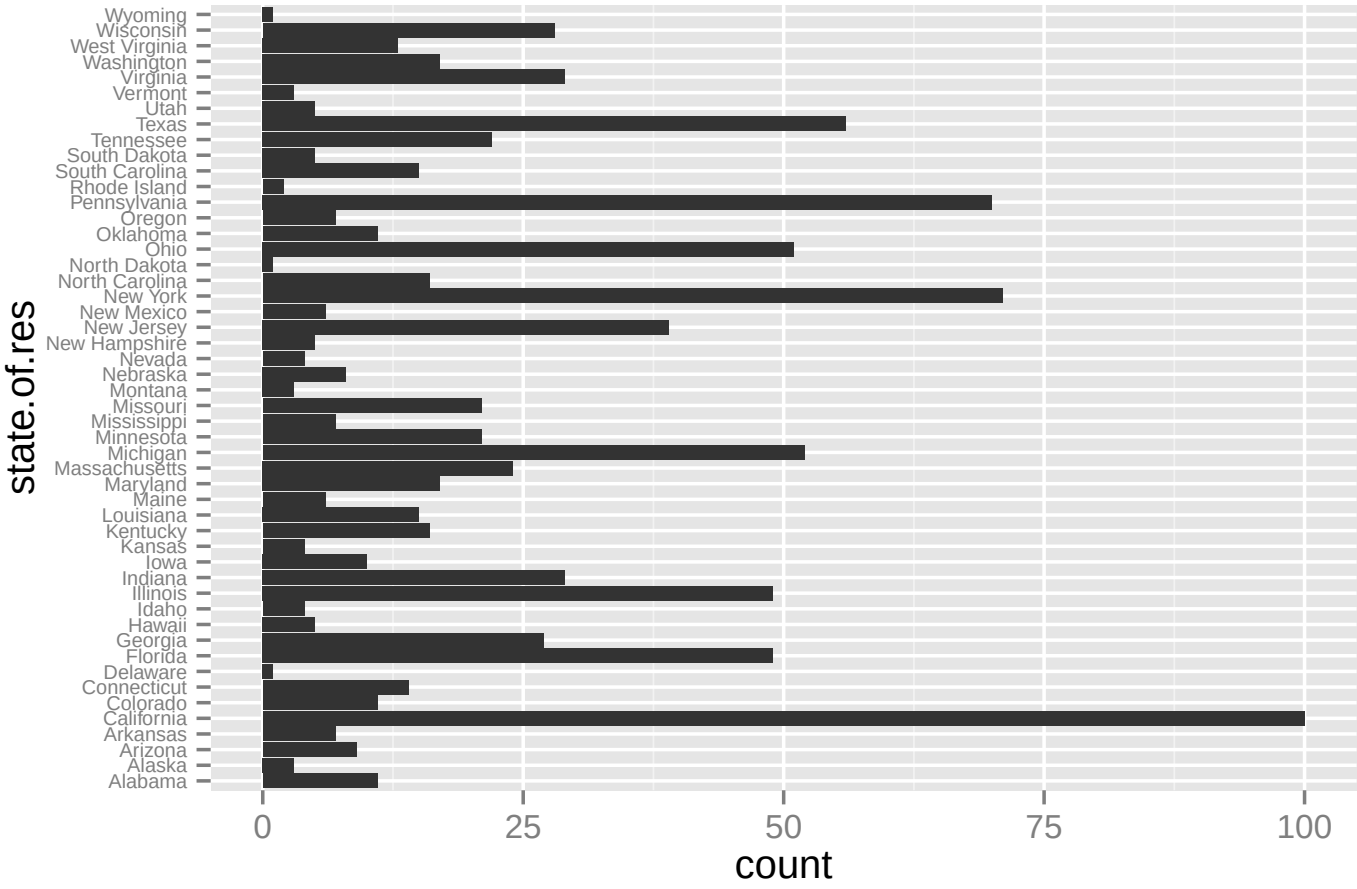
```
qplot(state.of.res, data=custdata) + coord_flip()
```



Better! When you have more than a few categories, use horizontal bars!

Bar charts

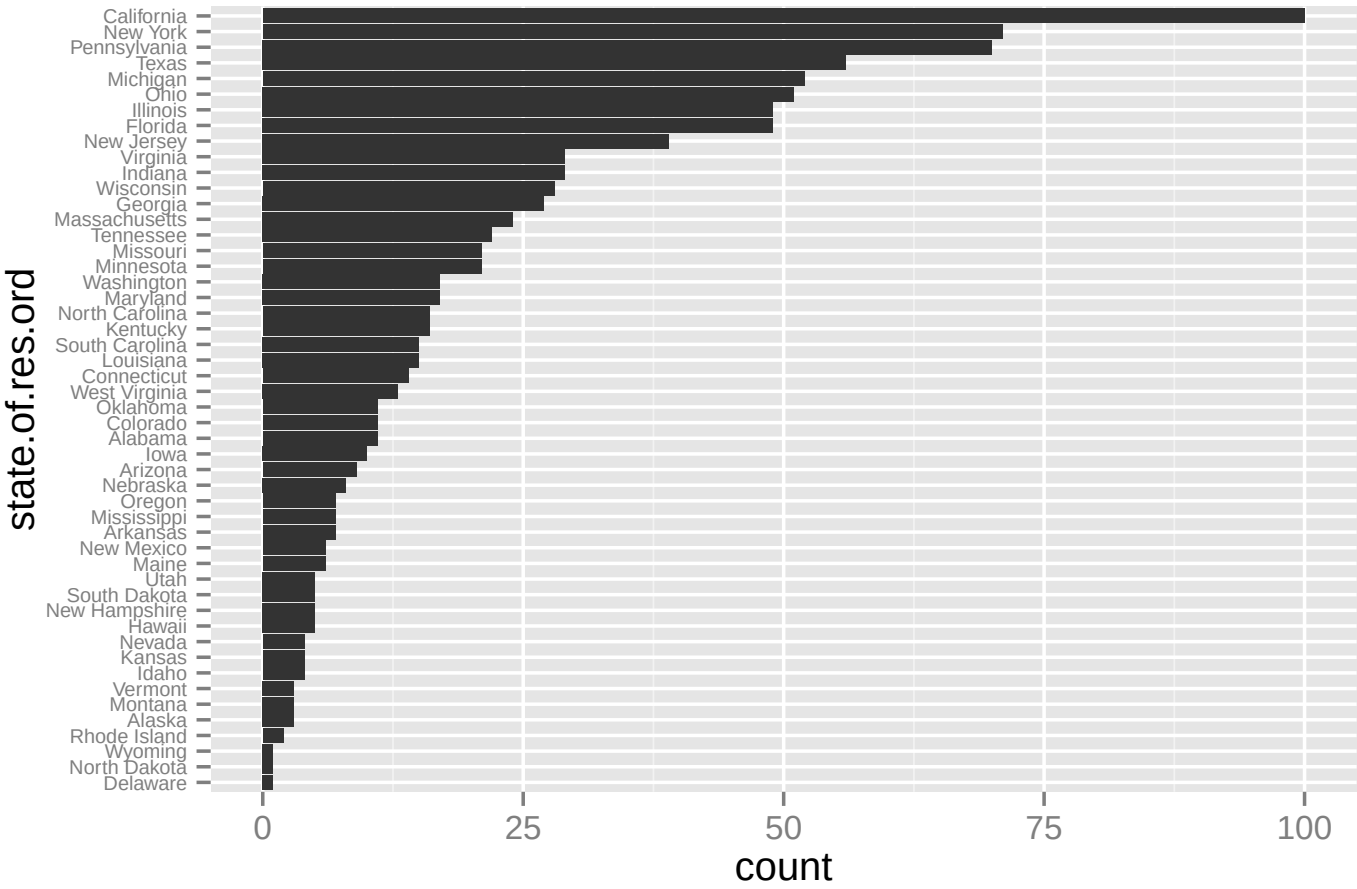
```
qplot(state.of.res, data=custdata) + coord_flip() +
    theme(axis.text.y=element_text(size=rel(0.6)))
```



Better yet! The labels are small, but at least they don't overlap.

Sorted bar chart

```
custdata <- transform(custdata, state.of.res.ord=
        reorder(state.of.res, state.of.res, length))
qplot(state.of.res.ord, data=custdata) + coord_flip() +
    theme(axis.text.y=element_text(size=rel(0.6)))
```



Aside: reorder a factor

Let's reorder states by average number of vehicles per customer.

```
state.by.num.vehicles <- reorder(custdata$state.of.res,
        custdata$num.vehicles, mean, na.rm=TRUE)
```

What is the average number of vehicles per customer in each state?

In Alabama:

```
with(custdata, mean(
    num.vehicles[state.of.res=="Alabama"], na.rm=TRUE
    ))
```

Repeat for each of the 50 states. There has to be a better way!

Aside: reorder a factor

Let's reorder states by average number of vehicles per customer.

```
state.by.num.vehicles <- reorder(custdata$state.of.res,
        custdata$num.vehicles, mean, na.rm=TRUE)
```

What is the average number of vehicles per customer in each state?

In Alabama:

```
with(custdata, mean(
    num.vehicles[state.of.res=="Alabama"], na.rm=TRUE
    ))
```

Repeat for each of the 50 states. There has to be a better way!

Aside

Using base R

```
# split
pieces <- split(custdata, custdata$state.of.res)

# apply
result <- lapply(pieces, function(p) data.frame(
    state.of.res=p$state.of.res[[1]],
    state.avg.vehicles=mean(p$num.vehicles, na.rm=TRUE)
  )
)

# combine
result <- do.call("rbind", result)
```

Aside

Package `plyr` implements split-apply-combine framework very neatly in a single function call.

```
library(plyr)
result <- ddply(
    custdata,              # dataframe
    "state.of.res",        # split-by variables
    summarize,             # function to apply to each piece
                           # function arguments
    state.avg.vehicles=mean(num.vehicles, na.rm=TRUE)
)
```

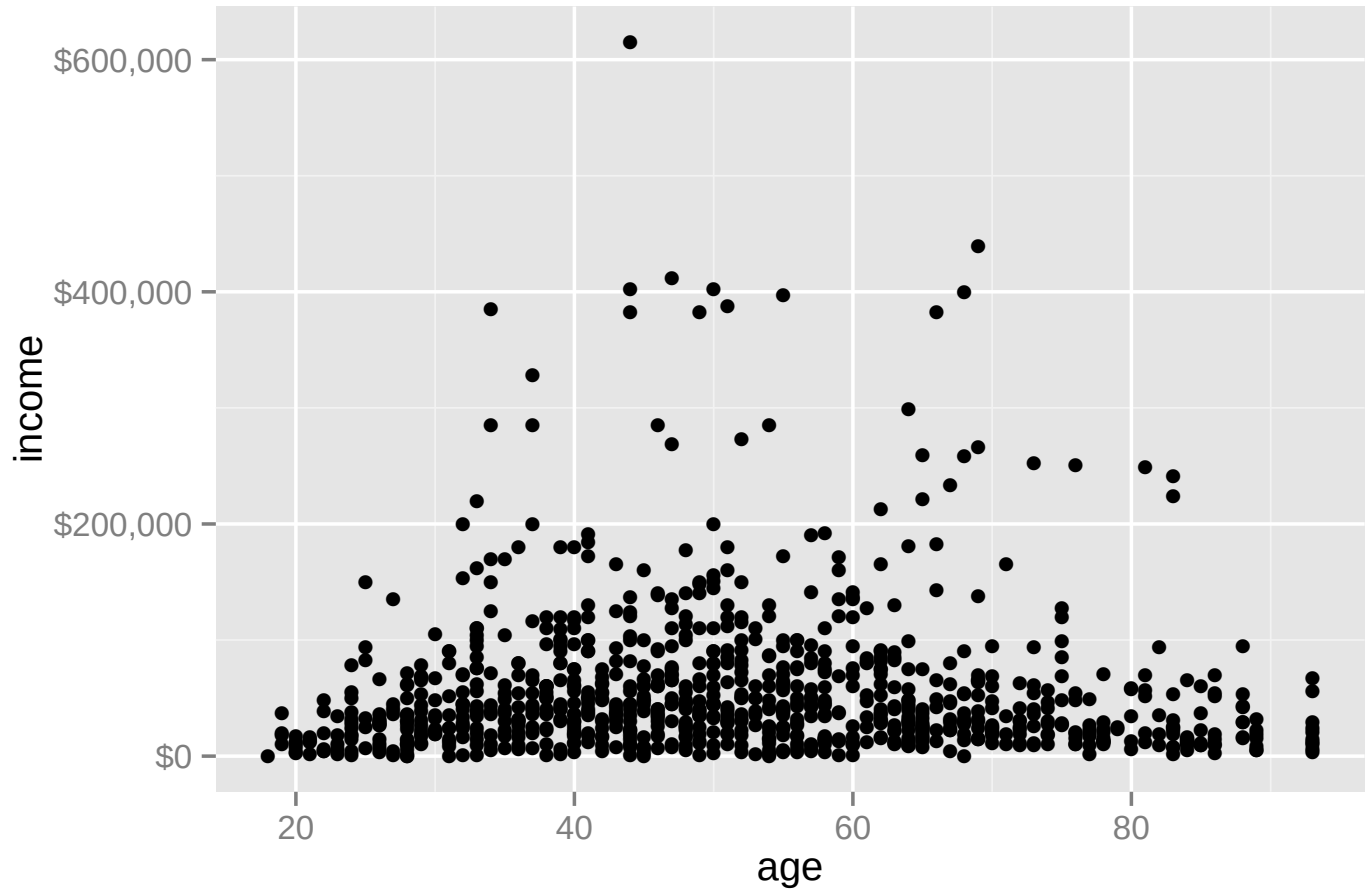
Single variable

To summarize visualization of single variable

- For a numerical variable use a histogram or density plot to look for outliers, or incorrect values.
- Also get a feel for the distribution – is it symmetric, normal, lognormal.
- For categorical variables use a bar chart to compare frequencies of categories.

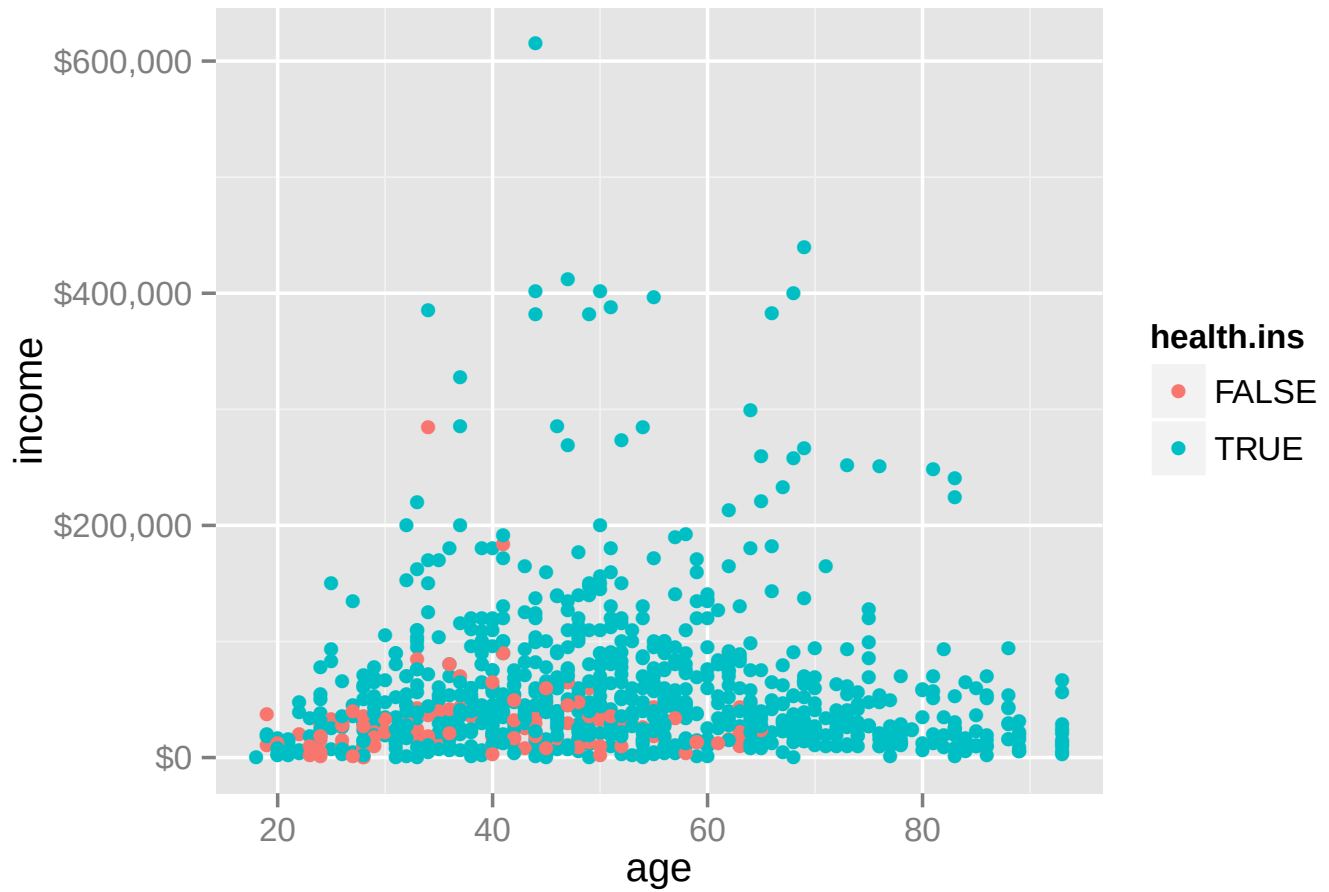
Scatter plot

```
custdata2 <- with(custdata,
    subset(custdata, age>0 & age < 100 & income > 0))
qplot(age, income, data=custdata2) +
    scale_y_continuous(labels=dollar)
```



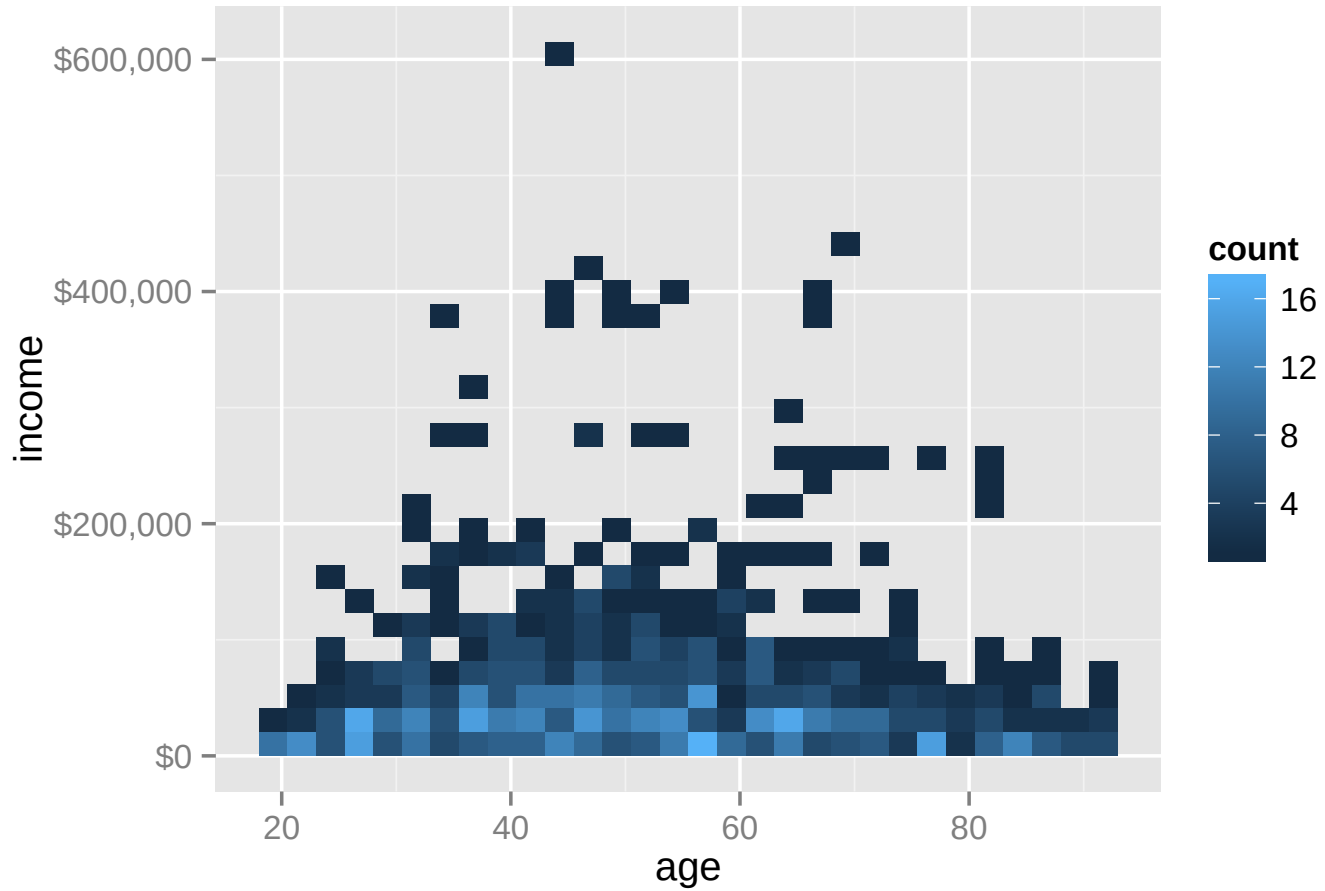
Scatter plot

```
qplot(age, income, data=custdata2, colour=health.ins) +
    scale_y_continuous(labels=dollar)
```



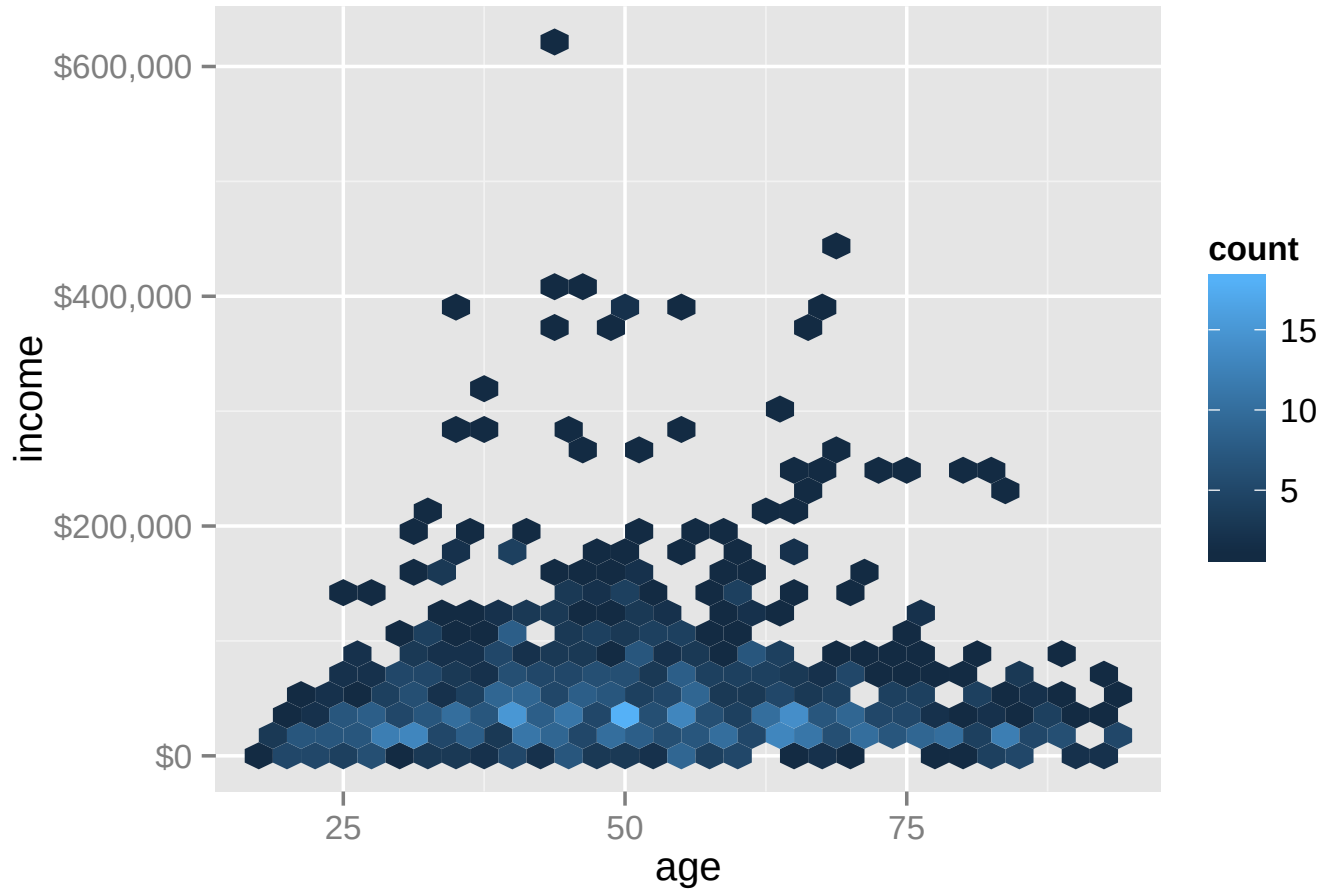
2D histogram

```
qplot(age, income, data=custdata2, geom="bin2d") +
    scale_y_continuous(labels=dollar)
```



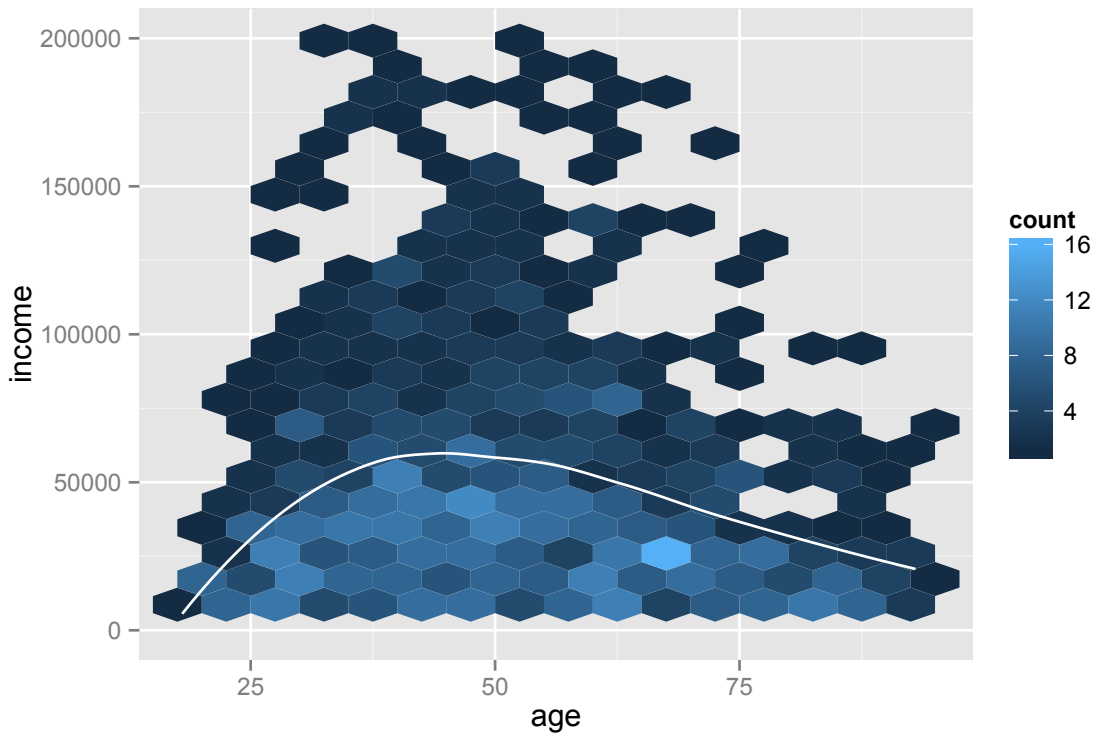
2D histogram

```
library(hexbin)
qplot(age, income, data=custdata2, geom="hex") +
    scale_y_continuous(labels=dollar)
```



2D histogram

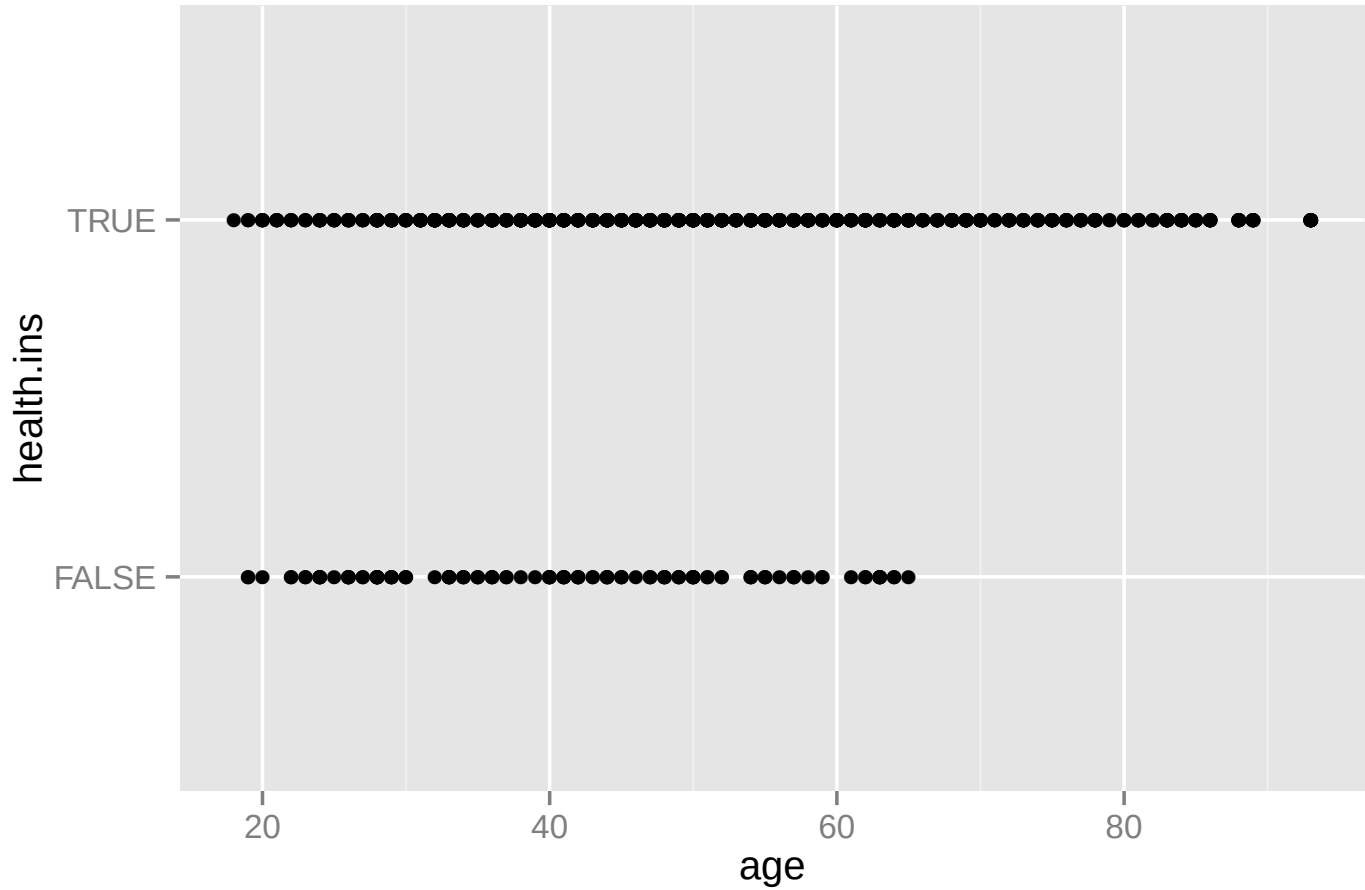
```
library(hexbin)
ggplot(custdata2, aes(x=age, y=income)) +
   geom_hex(binwidth=c(5, 10000)) +
   geom_smooth(color="white", se=F) +
   ylim(0,200000)
```



Scatter plot

Also works for continuous vs. categorical.

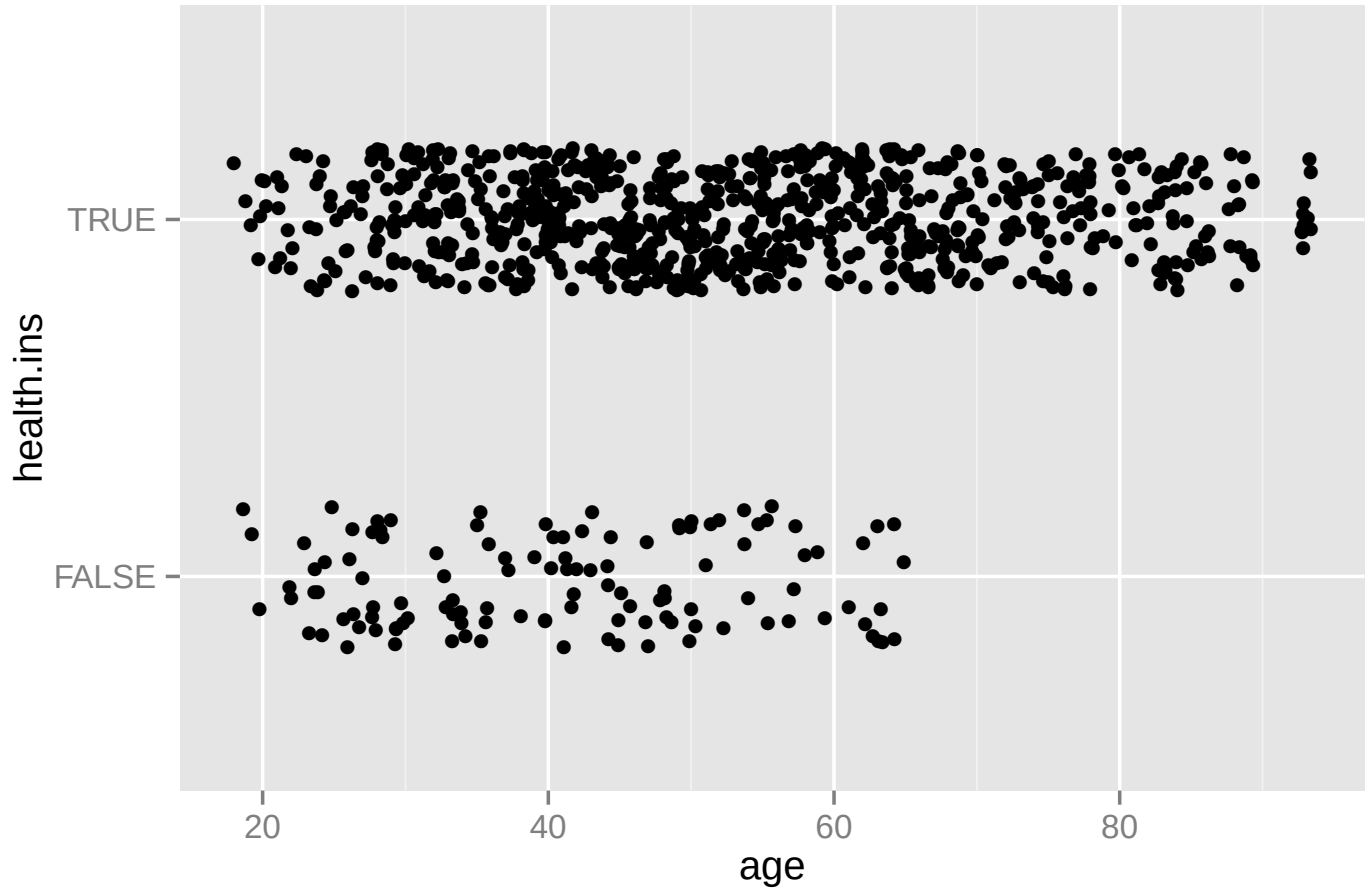
```
qplot(age, health.ins, data=custdata2)
```



Scatter plot

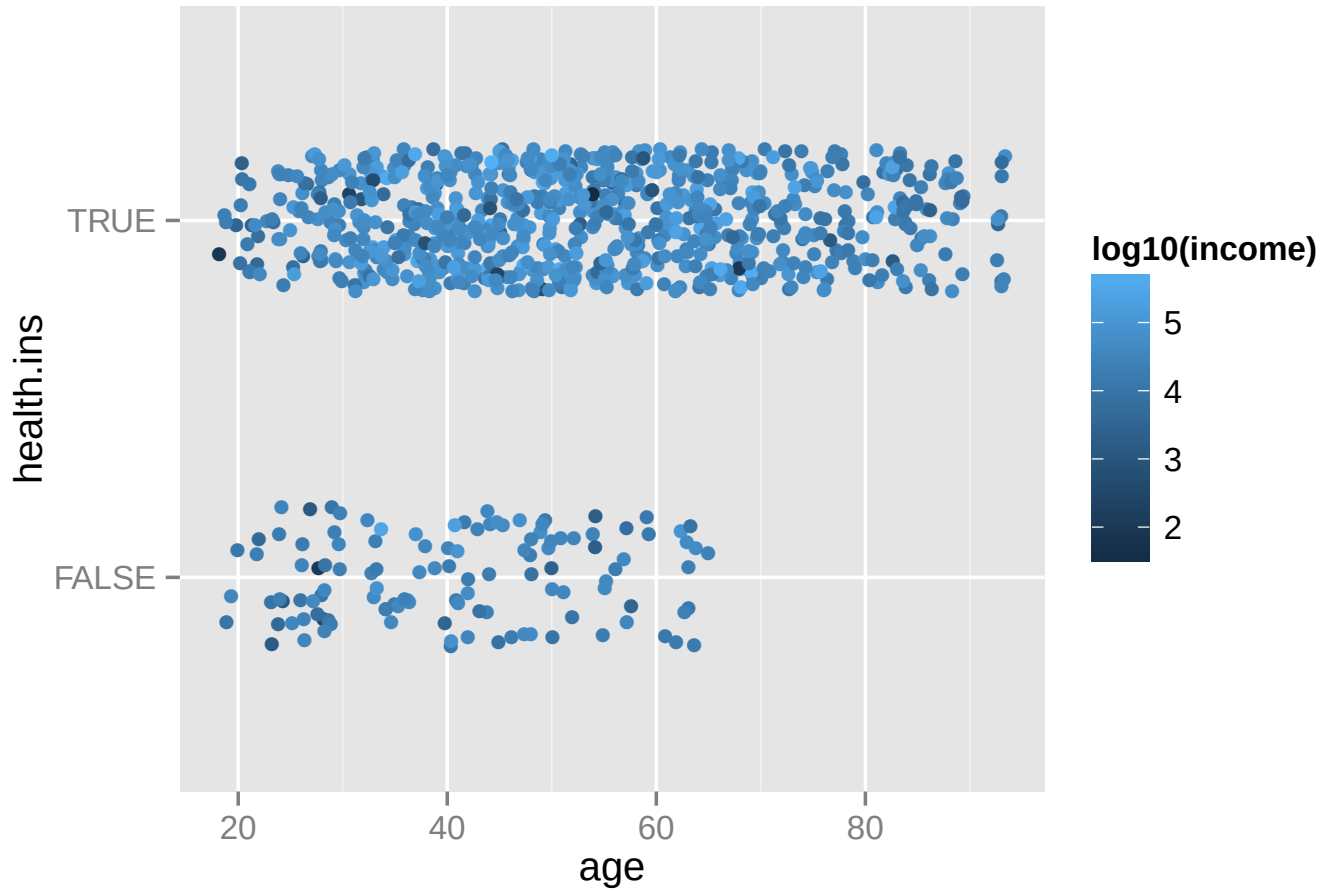
This is better – it gives a better feel for the density at each level.

```
qplot(age, health.ins, data=custdata2,
    position=position_jitter(height=0.2))
```



Scatter plot

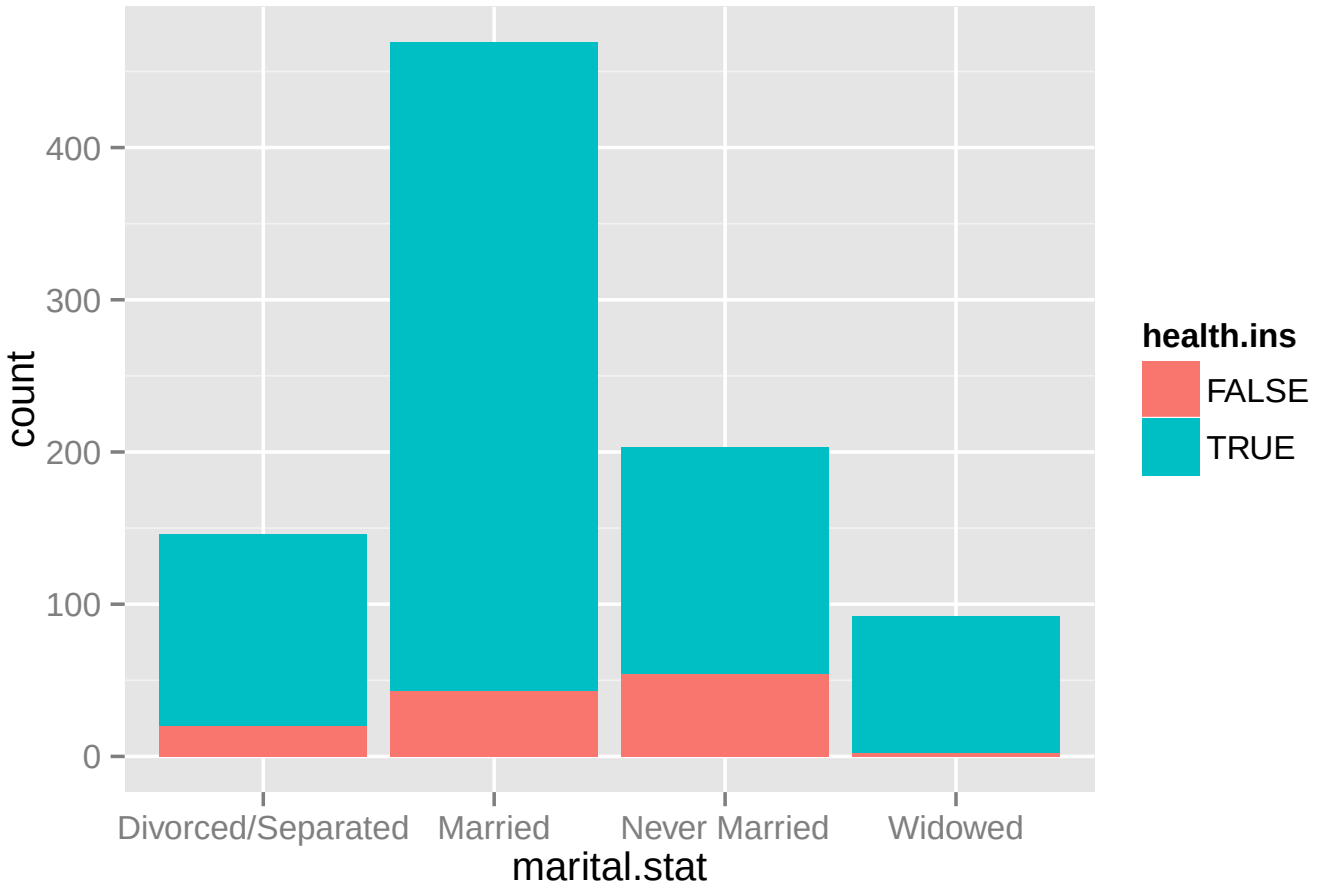
```
qplot(age, health.ins, data=custdata2, color=log10(income),
    position=position_jitter(height=0.2))
```



Bar chart for two variables

Use the `fill` aesthetic as the second variable

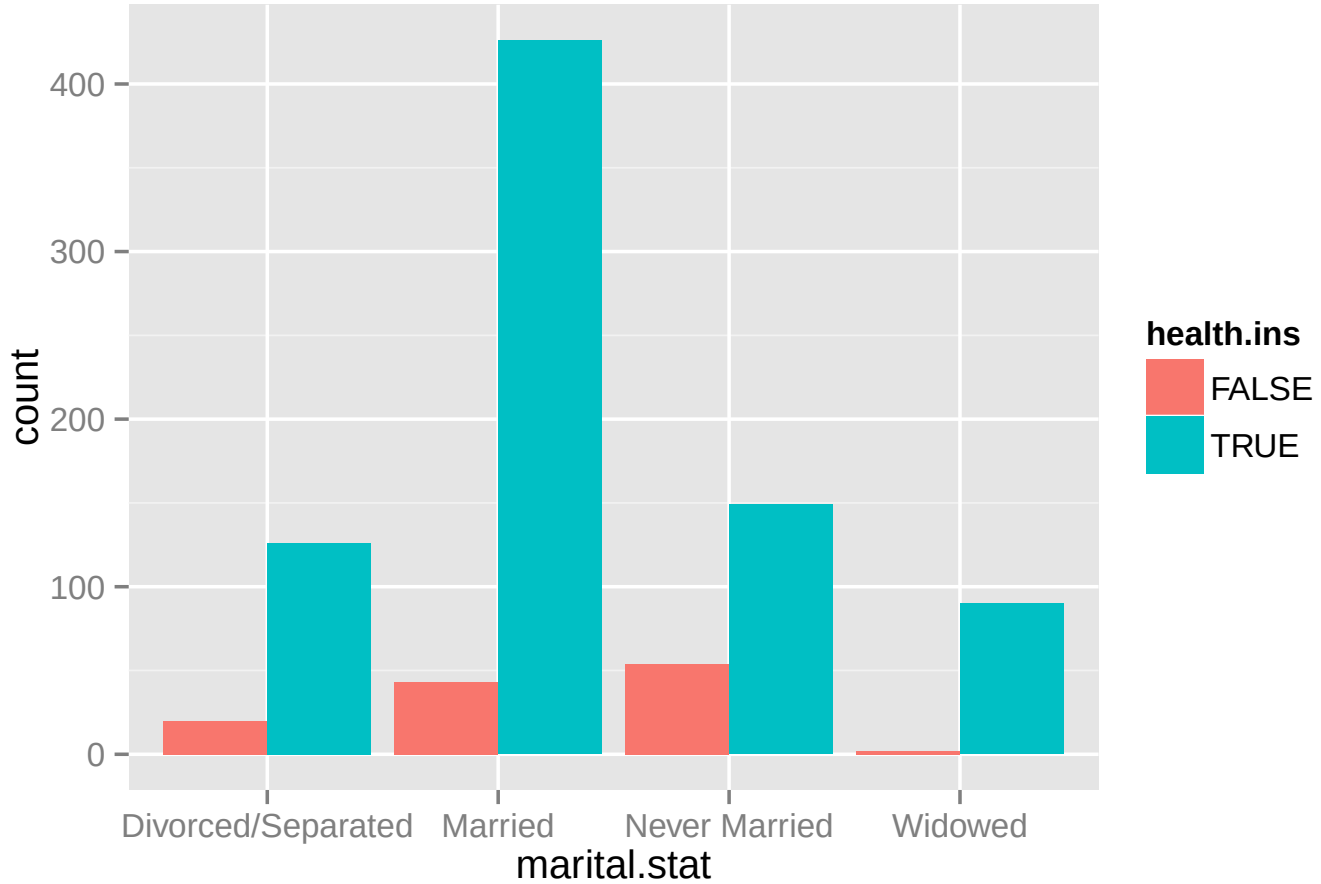
```
qplot(marital.stat, data=custdata2, fill=health.ins)
```



Bar chart for two variables

Some prefer side-by-side

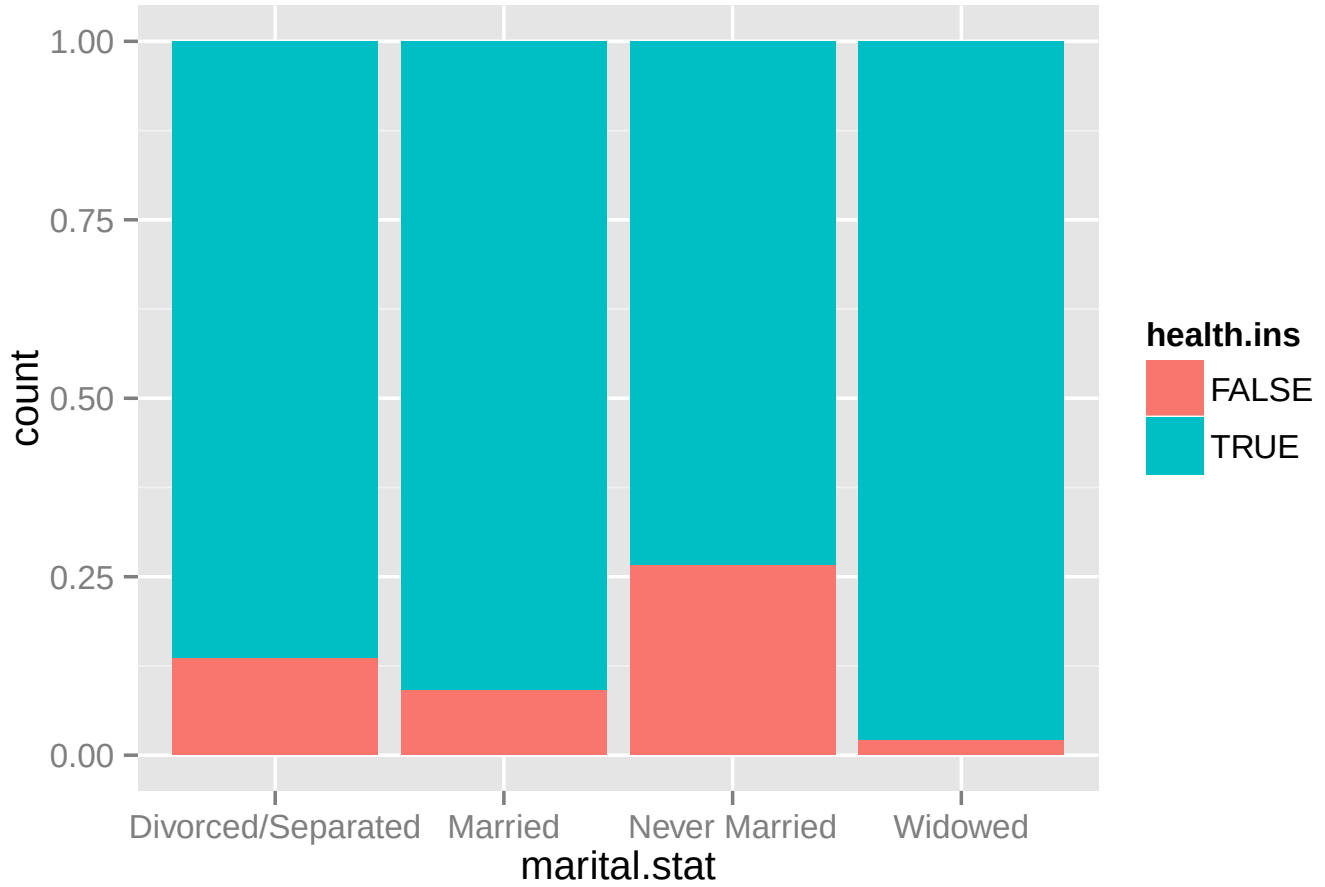
```
ggplot(custdata2) +
    geom_bar(aes(marital.stat, fill=health.ins),
             position="dodge")
```



Bar chart for two variables

Filled bar chart shows the proportion of insured within each level of marital status.

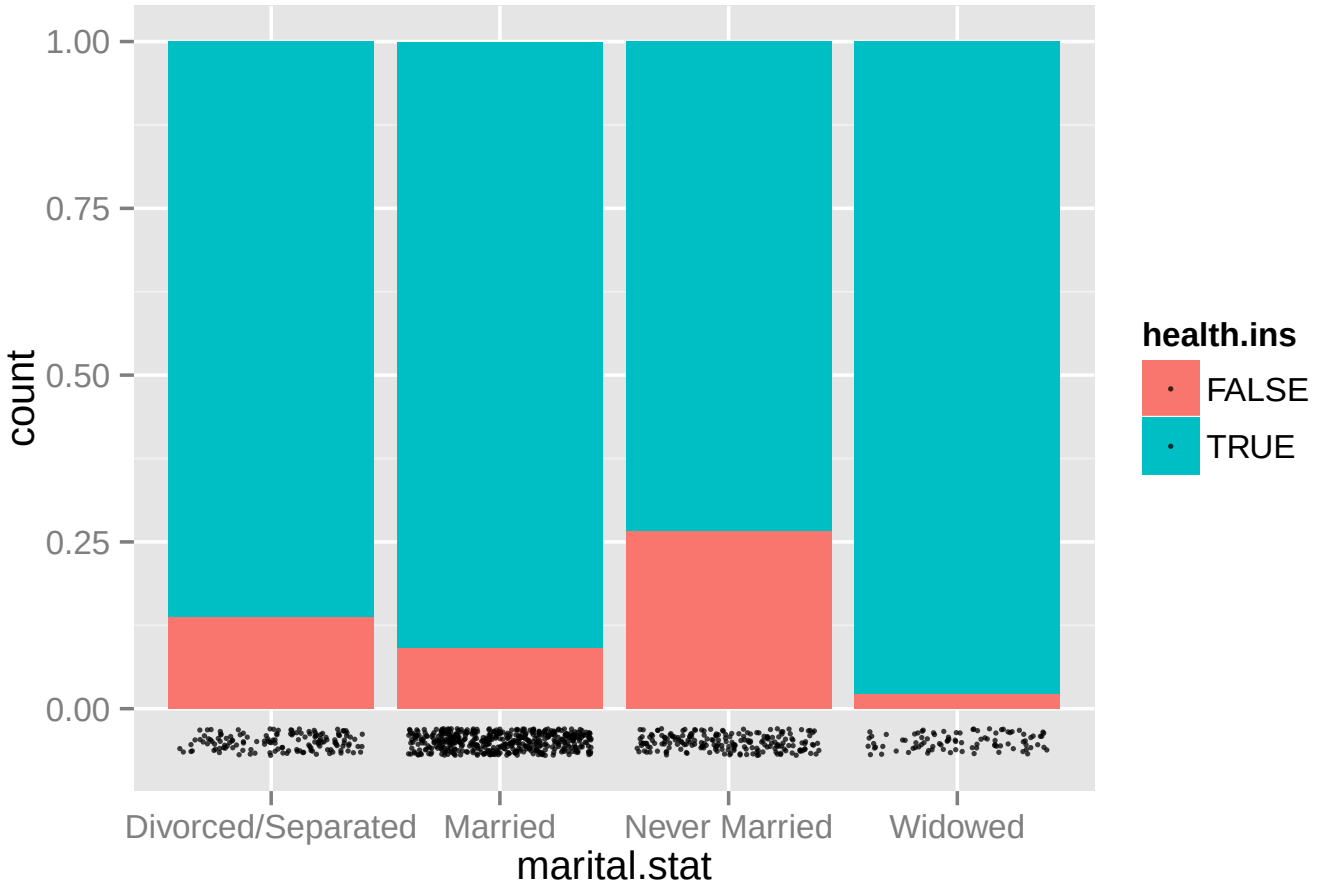
```
ggplot(custdata2) +
    geom_bar(aes(marital.stat, fill=health.ins),
             position="fill")
```



Bar chart for two variables

Add a cloud of points to convey the size of each level.

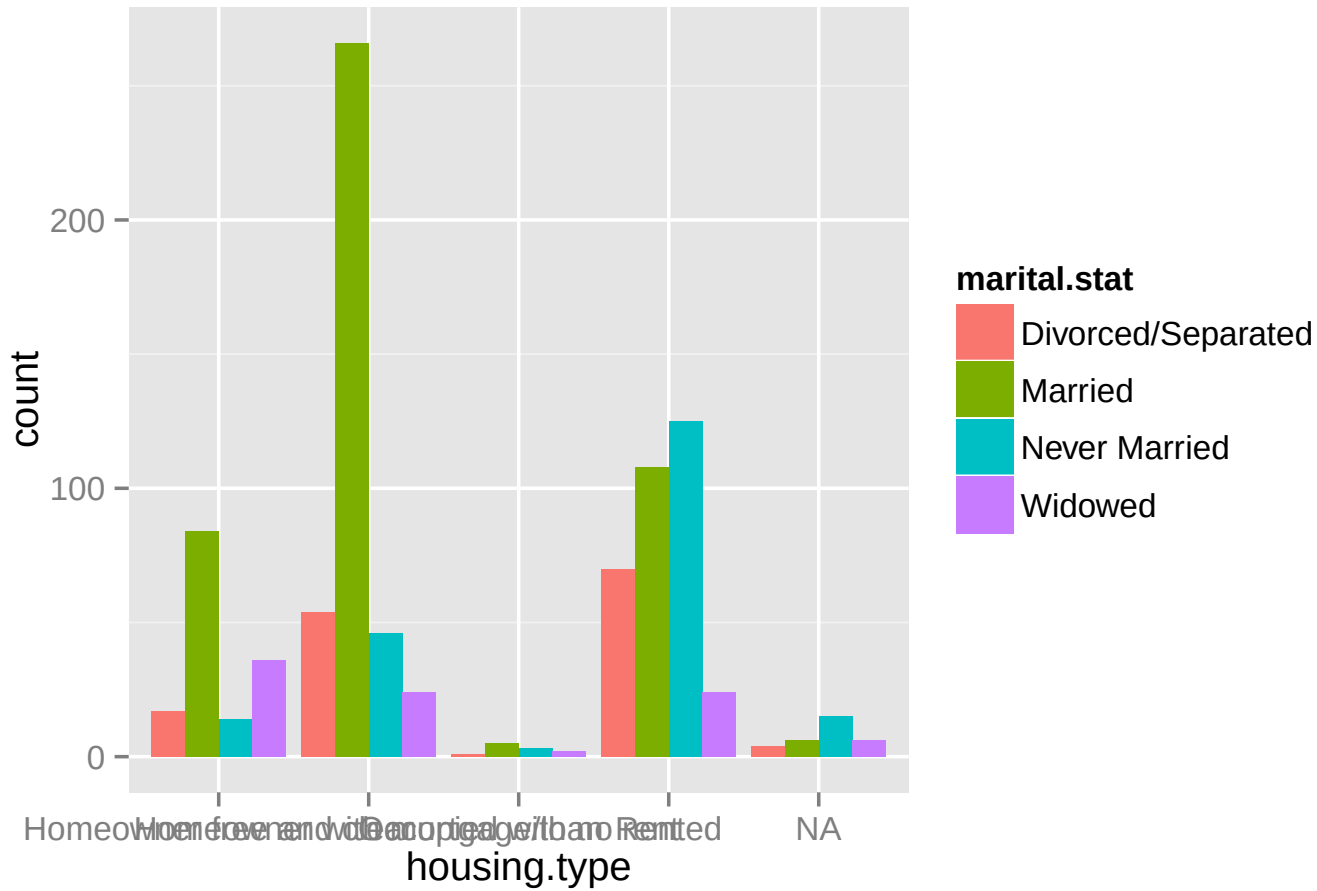
```
last_plot() + geom_point(aes(x=marital.stat, y=-0.05),
    position=position_jitter(h=0.02), size=0.75, alpha=0.75)
```



Bar chart for two variables

More than two levels

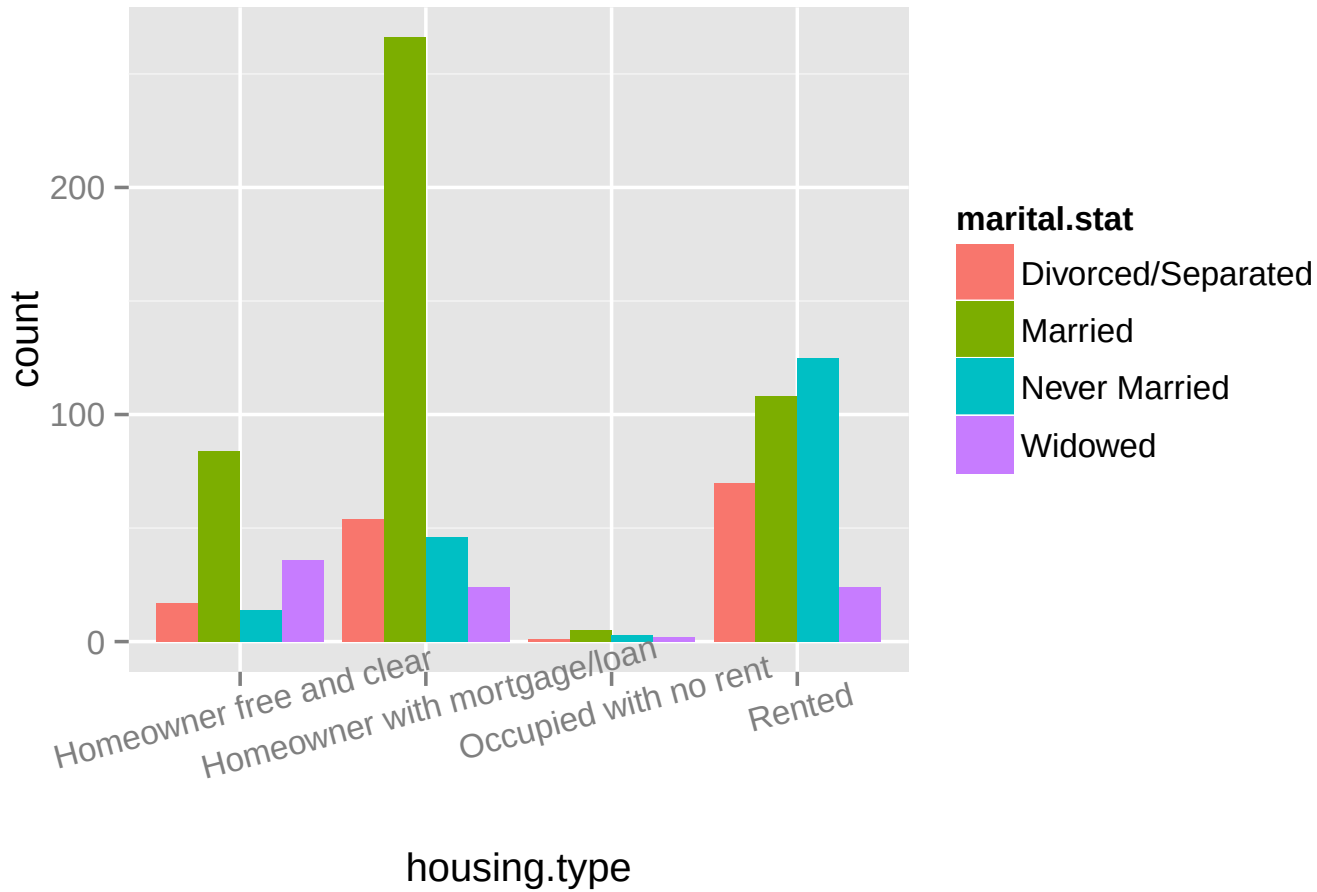
```
ggplot(custdata2) +
    geom_bar(aes(housing.type, fill=marital.stat),
             position="dodge")
```



Bar chart for two variables

Remove `NA` from housing.type and fix labels

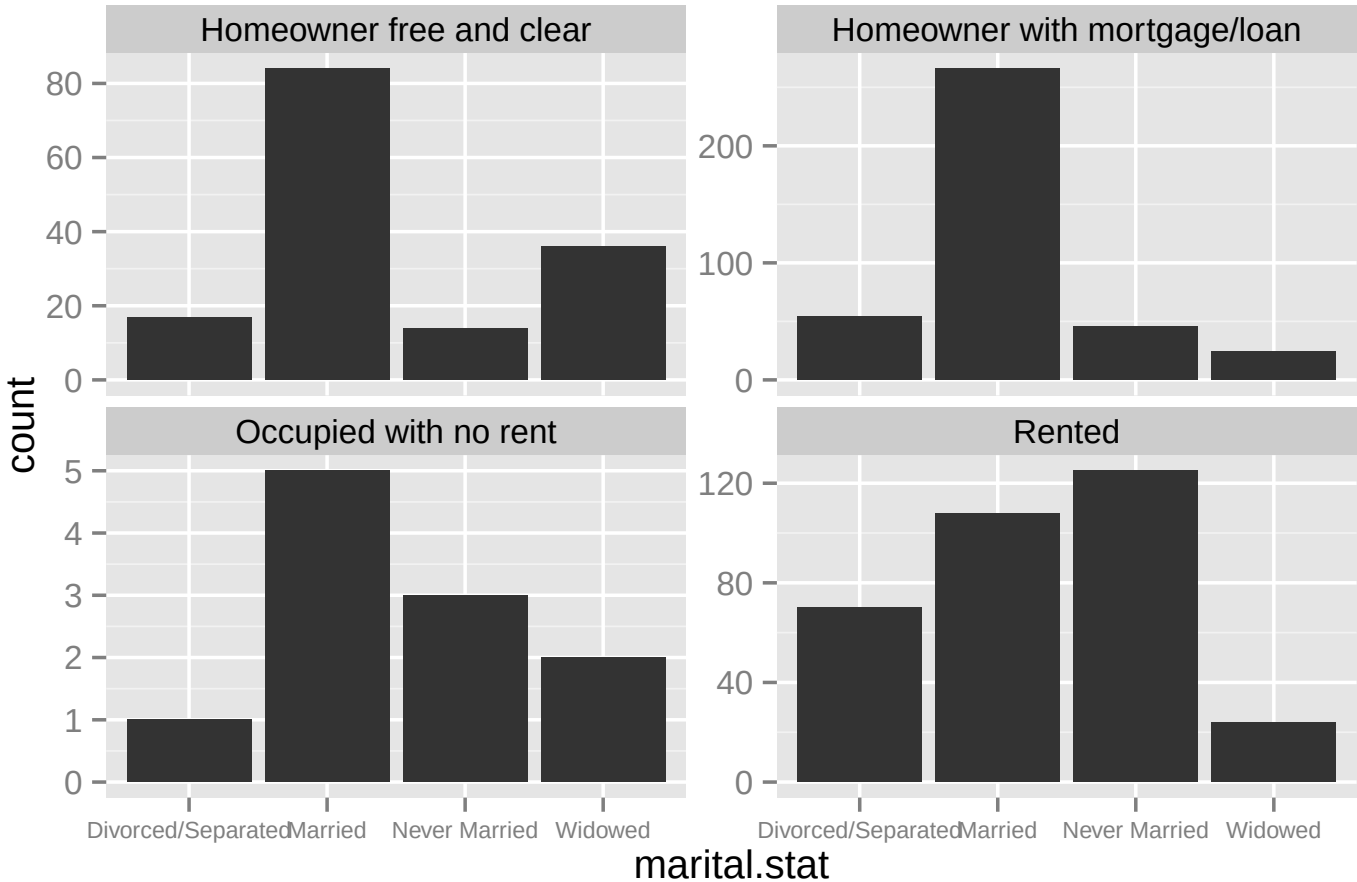
```
ggplot(subset(custdata2, !is.na(housing.type))) +
    geom_bar(aes(housing.type, fill=marital.stat),
             position="dodge") +
    theme(axis.text.x=element_text(angle=15))
```



Bar chart for two variables

Use facetting instead of fill to get a better picture.

```
ggplot(subset(custdata2, !is.na(housing.type))) +
    geom_bar(aes(marital.stat)) +
    facet_wrap(~housing.type, scales="free_y") +
    theme(axis.text.x=element_text(size=rel(0.8)))
```



Visualization with R

Further readings

- A short course on `ggplot2` by Hadley Wickham
  `http://courses.had.co.nz/11-rice/`