Introduction to R Workshop (Session 2)

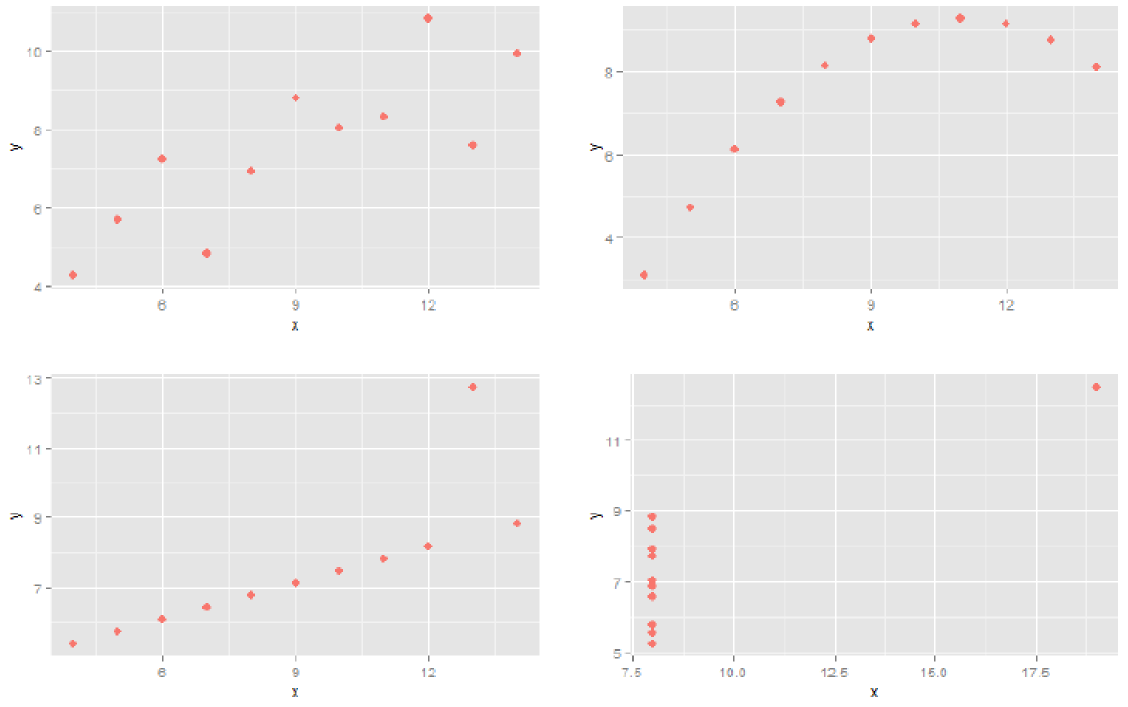
Dr Carlos Moreno-Garcia

School of Computing

Robert Gordon University

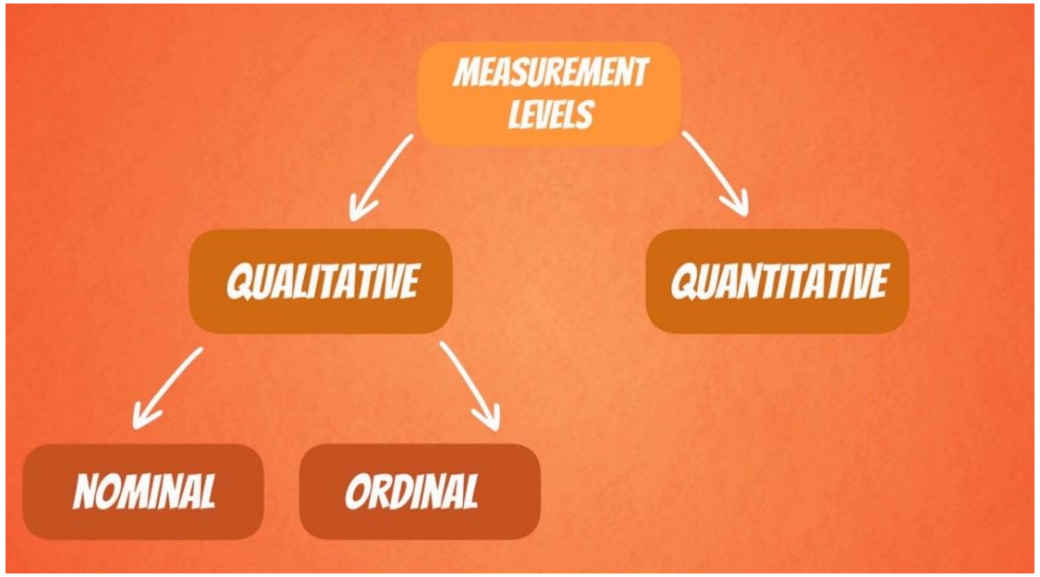
Data Visualisation

- Exploratory data analysis.
- Understand data properties.
- Find patterns in data.
- Suggest modelling strategies.
- Verify analysis.
- Communicate results.



Data Types

- Nominal: Categorical data.
  - e.g. gender (Factors)
- Ordinal: Categorical with a logical order.
  - e.g. marks (Ordered Factors)
- Quantitative: Numeric.
  - e.g. physical measurements



Plotting Systems in R: Base

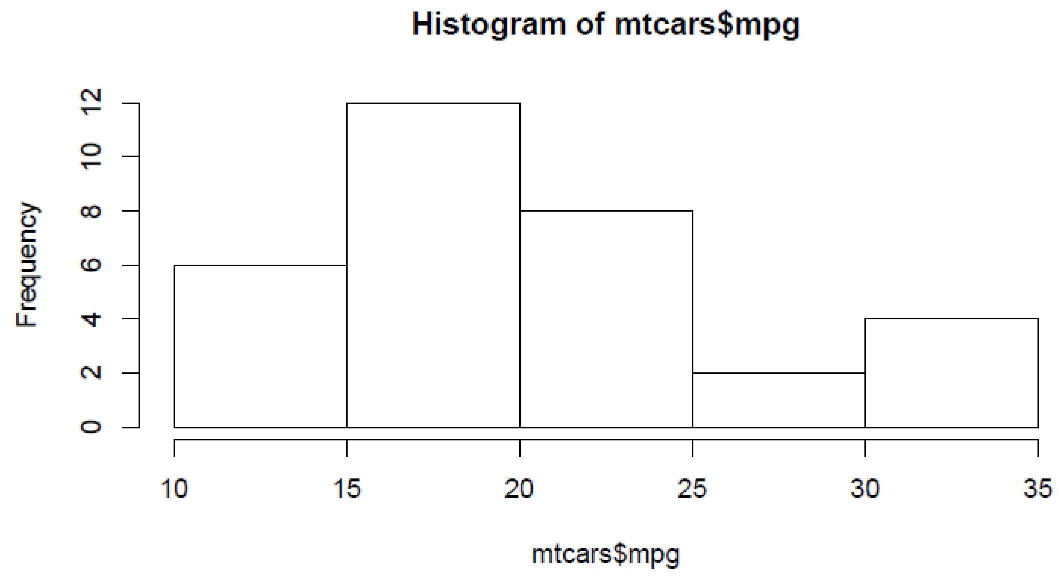
- “Artist’s palette” model.
- Start with blank canvas and build up from there.
- Start with plot function (or similar).
- Use annotation functions to add/modify (text, lines, points, axis).

Pros

- Convenient, mirrors how we think of building plots and analysing data.

Cons

- Can’t go back once plot has started (i.e. to adjust margins).
- Need to plan in advance.
- Difficult to “translate” to others once a new plot has been created (no graphical “language”). Plot is just a series of R commands.



Plotting Systems in R: *lattice*

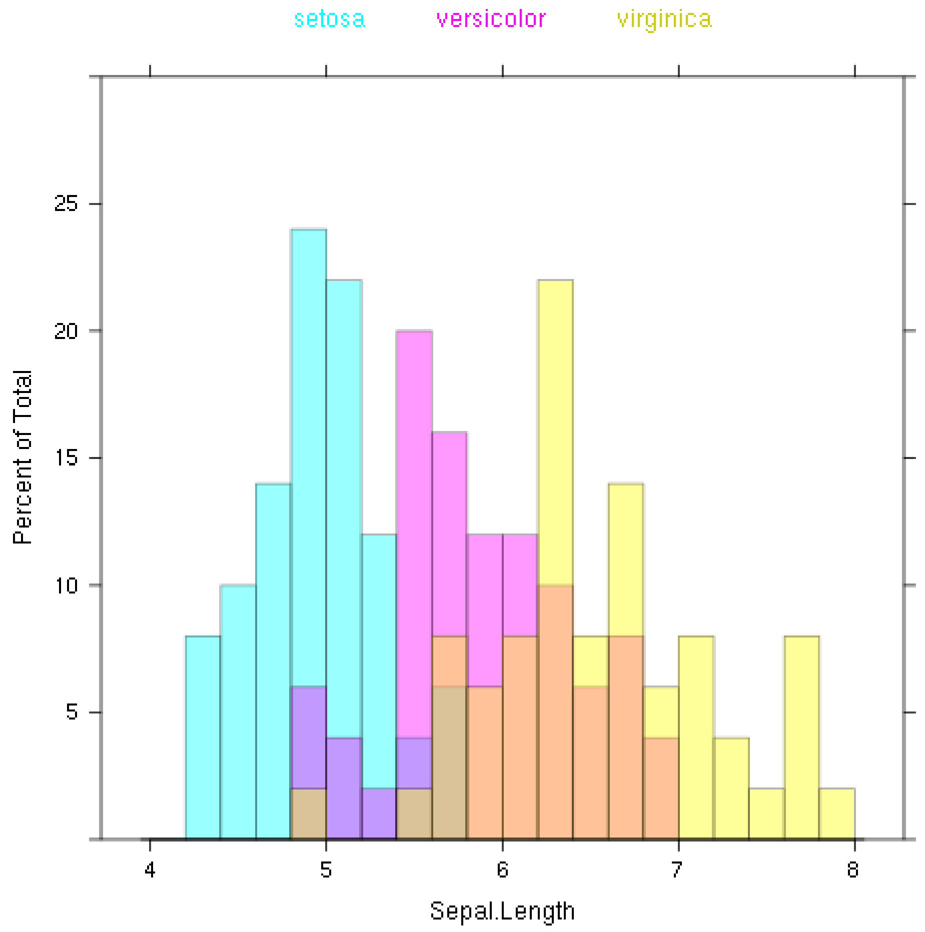
- Plots are created with a single function call (xyplot, bwplot, etc.)

Pros

- Most useful for conditioning types of plots: Looking at how y changes with x across levels of z.
- Things like margins/spacing set automatically because entire plot is specified at once.
- Good for putting many plots on a screen.

Cons

- Sometimes awkward to specify an entire plot in a single function call.
- Annotation in plot is not intuitive.
- Use of panel functions and subscripts difficult to wield and requires intense preparation.
- Cannot "add" to the plot once it's created.



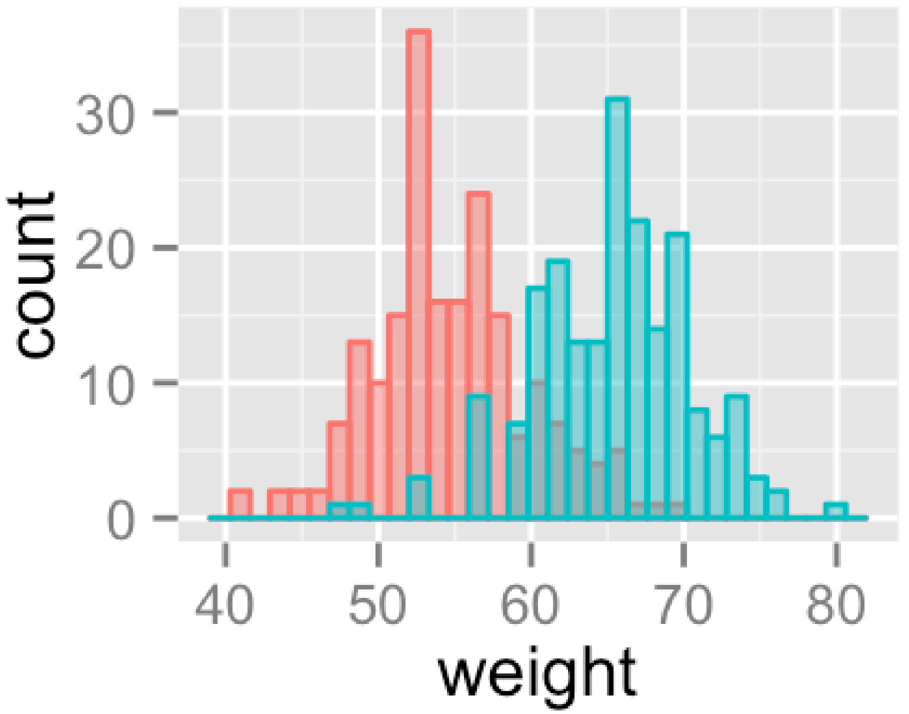
https://stackoverflow.com/questions/2759556/r-what-are-the-pros-and-cons-of-using-lattice-versus-ggplot2

Plotting Systems in R: *ggplot*

- A graphical module that you can install in R.

Pros

- Split the difference between base and lattice.
- Automatically deals with spacing, text, titles but also allows you to annotate by “adding”.
- Superficial similarity to lattice but generally easier/more intuitive to use.
- Default mode makes many choices for you (but you can customize).



https://stackoverflow.com/questions/2759556/r-what-are-the-pros-and-cons-of-using-lattice-versus-ggplot2

ggplot syntax

- Install the “tidyverse” package
  - install.packages(‘tidyverse’)

- Import the ggplot library
  - library(ggplot2)

- Initialise a ggplot object
  - ggplot(data, aes(x,y,…))
    - data is a data frame
    - aes() specifies the options that apply to all layers of the plot

- Add a geometry layer
  - + geom_<geometry_type>() e.g. histogram, scatterplot, etc.

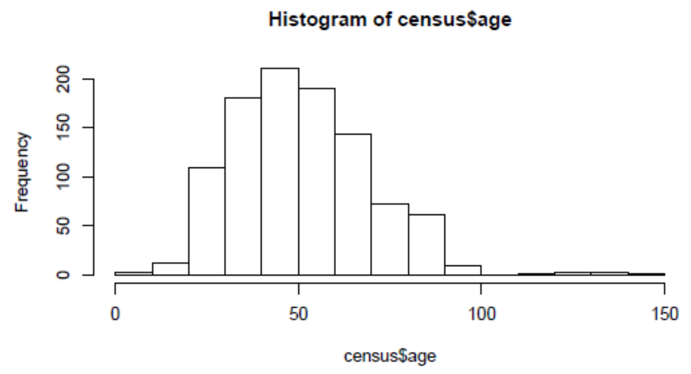
- Add other layers
  - + labs, + facet_wrap, etc.

Exploring a Single Variable

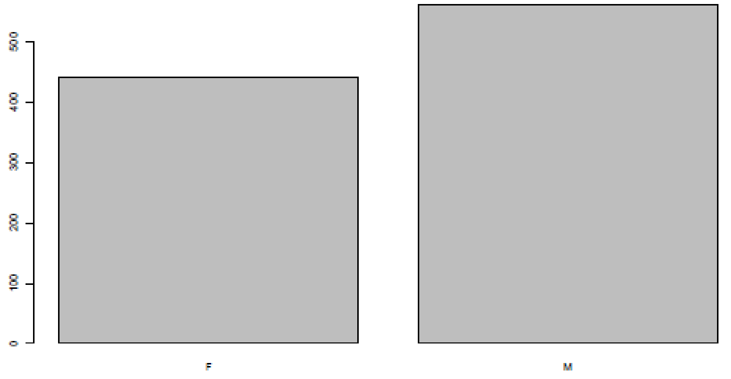
Exploring a Single Variable

- Useful for understanding the distribution (spread) of values of a variable.
- Can reveal any skewness in the data.
- Can help to identify out-of-range values.

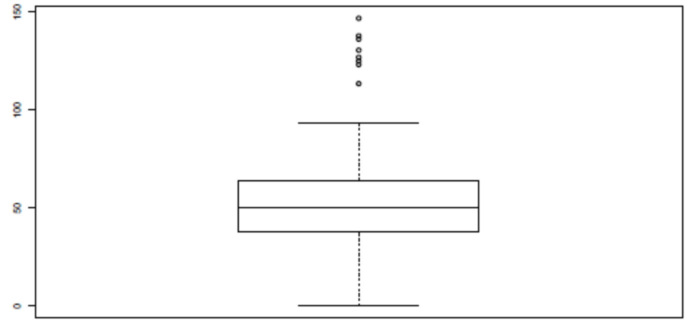
```
> hist(census$age)
```



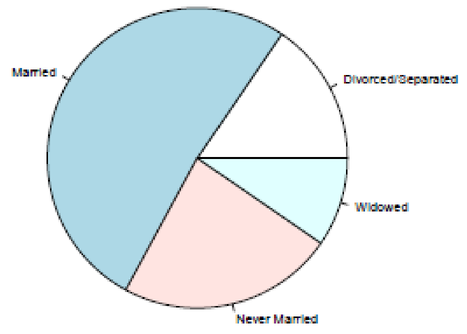
```
> barplot(table(census$sex))
```



```
> boxplot(census$age)
```

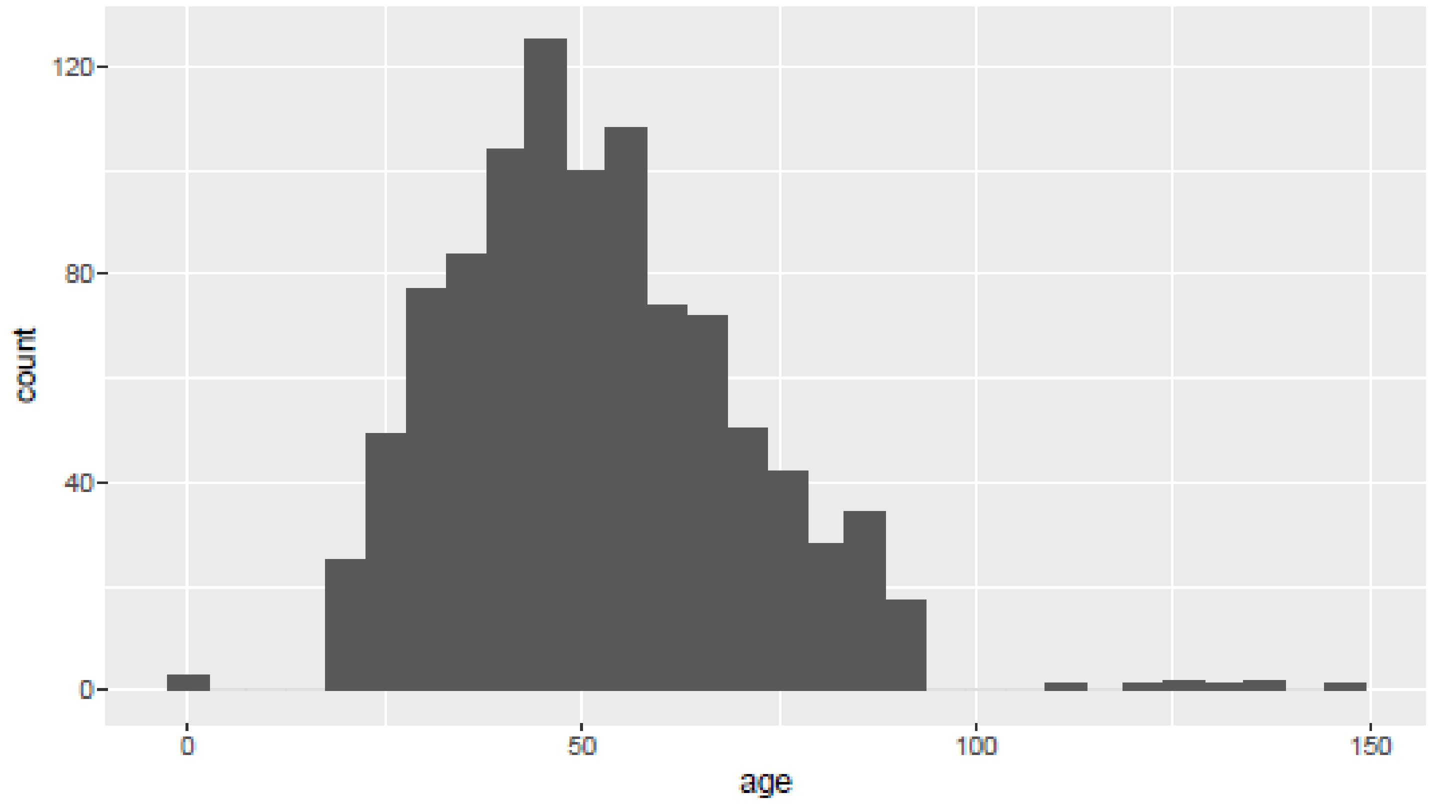


```
> pie(table(census$marital.stat))
```



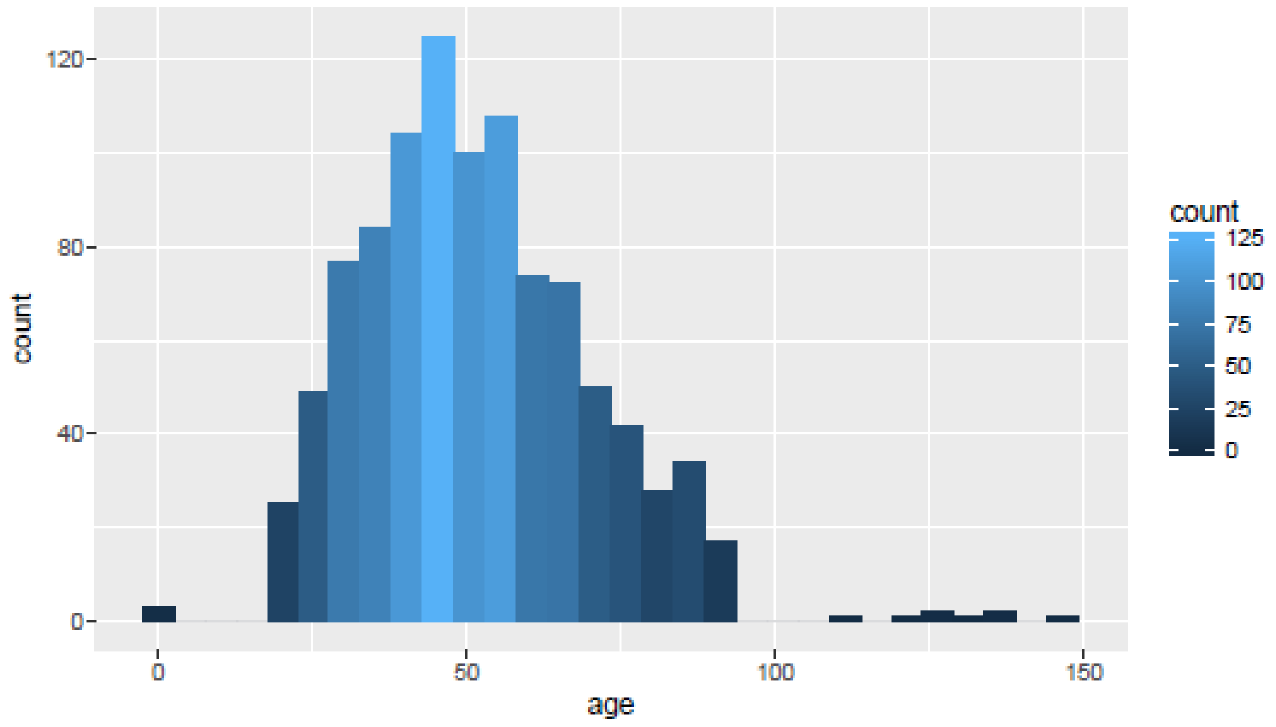
Histograms in *ggplot*

```
> ggplot(data=census, aes(x=age)) + geom_histogram()
```



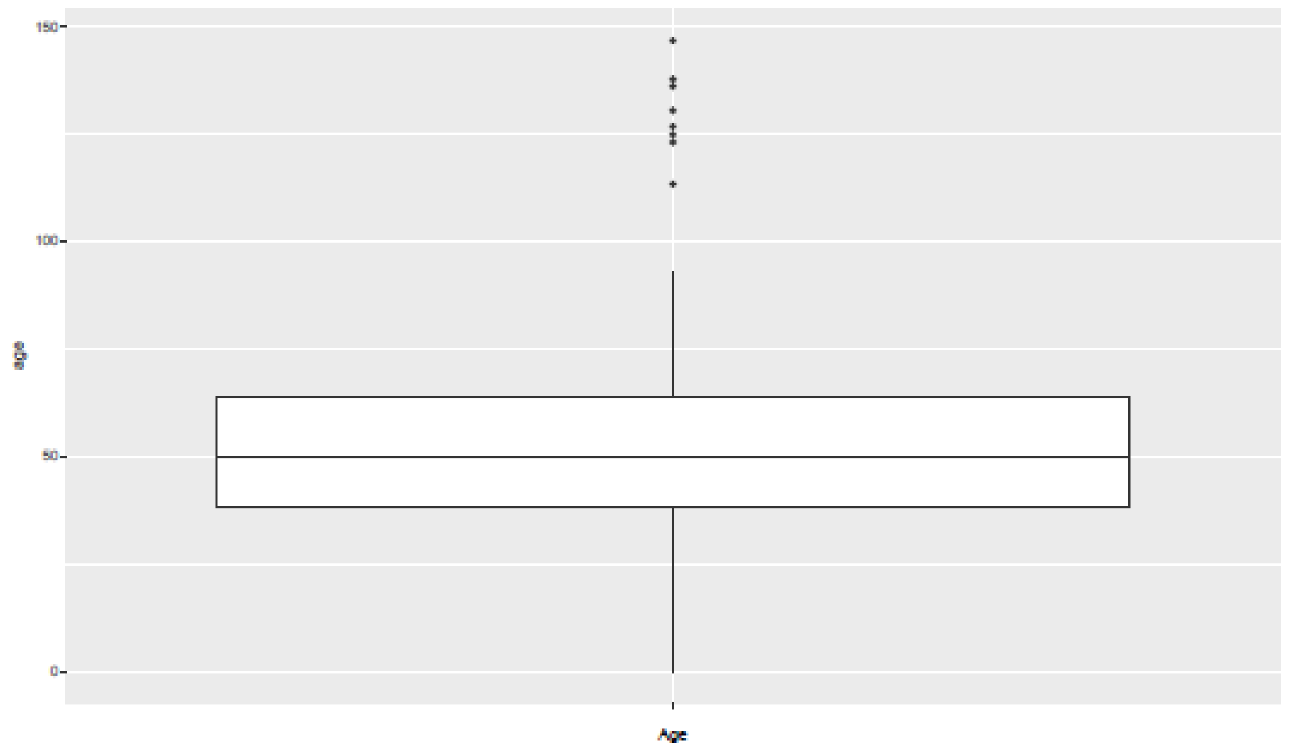
Histograms in *ggplot*

```
> ggplot(data=census, aes(x=age)) +  geom_histogram(aes(fill=..count..))
```



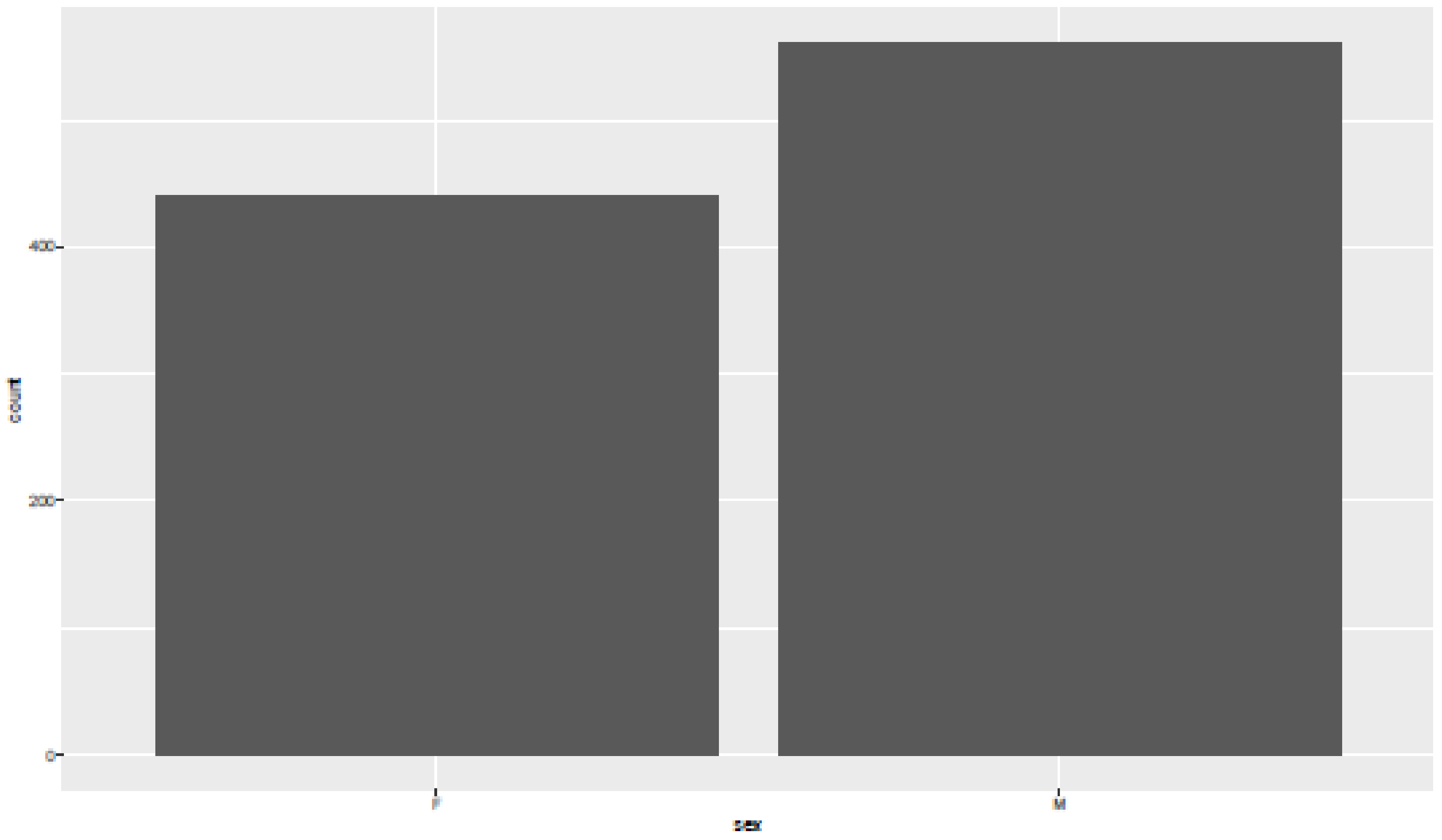
Boxplots in *ggplot*

```
> ggplot(data=census, aes(y=age, x="")) + geom_boxplot() + labs(x="Age")
```



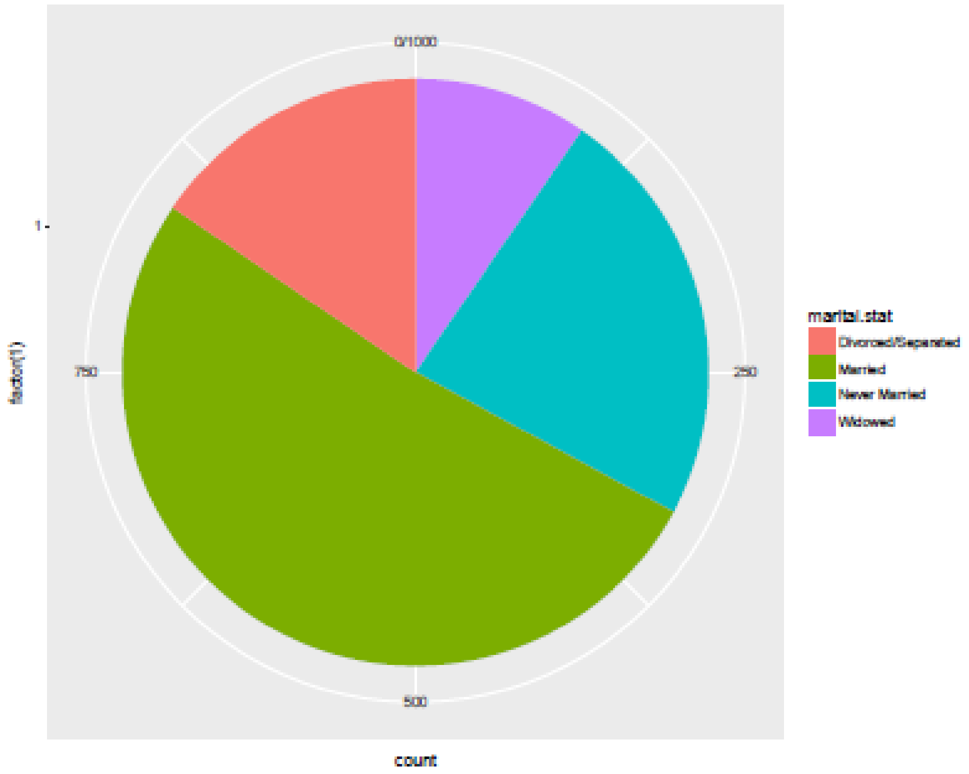
Bar plots in *ggplot*

```
> ggplot(census, aes(sex)) + geom_bar()
```



Pie charts in *ggplot*

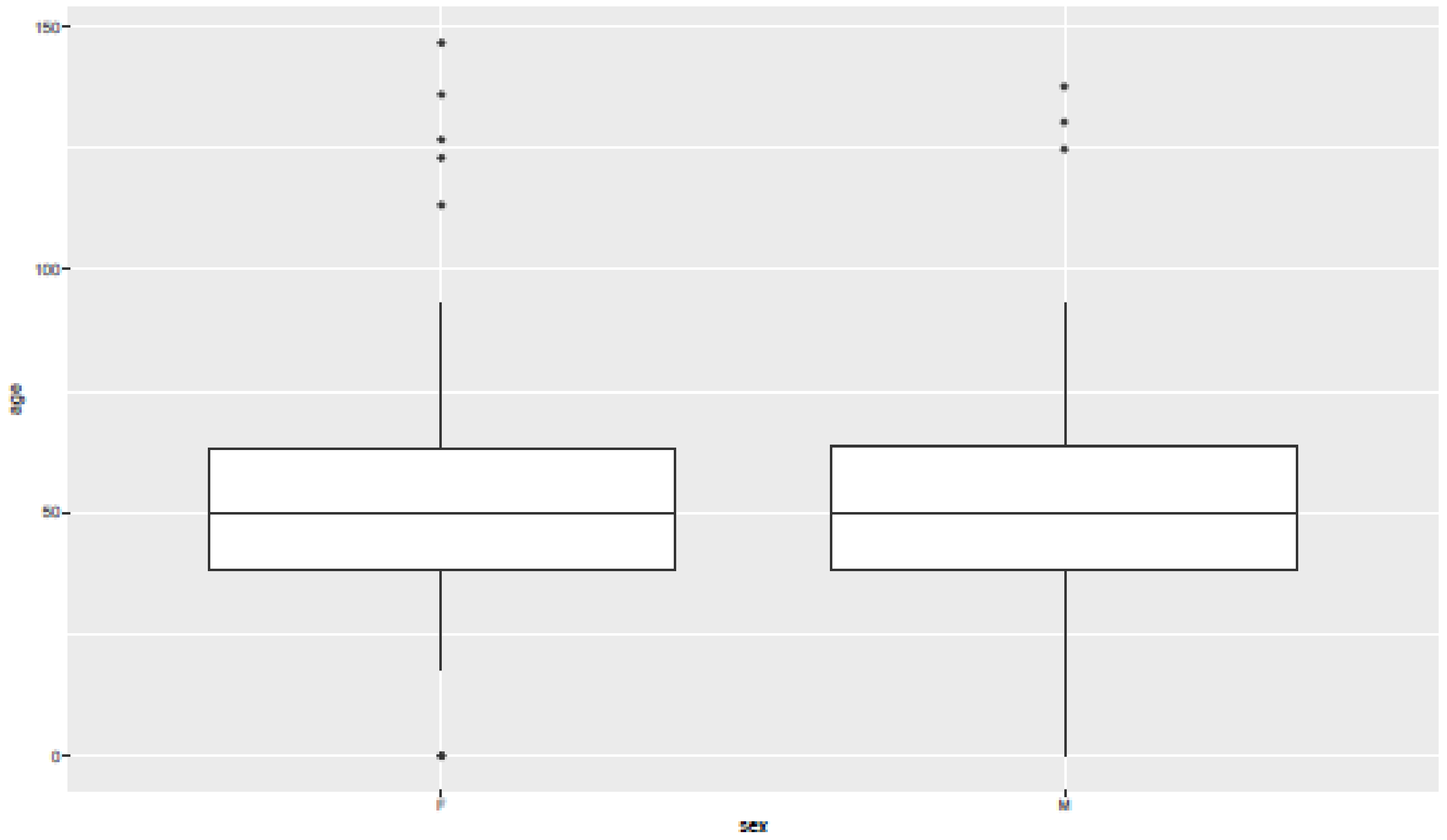
```
> ggplot(census, aes(factor(1), fill=marital.stat)) +
+       geom_bar(width=1) + coord_polar(theta="y")
```



Exploring Multiple Variables

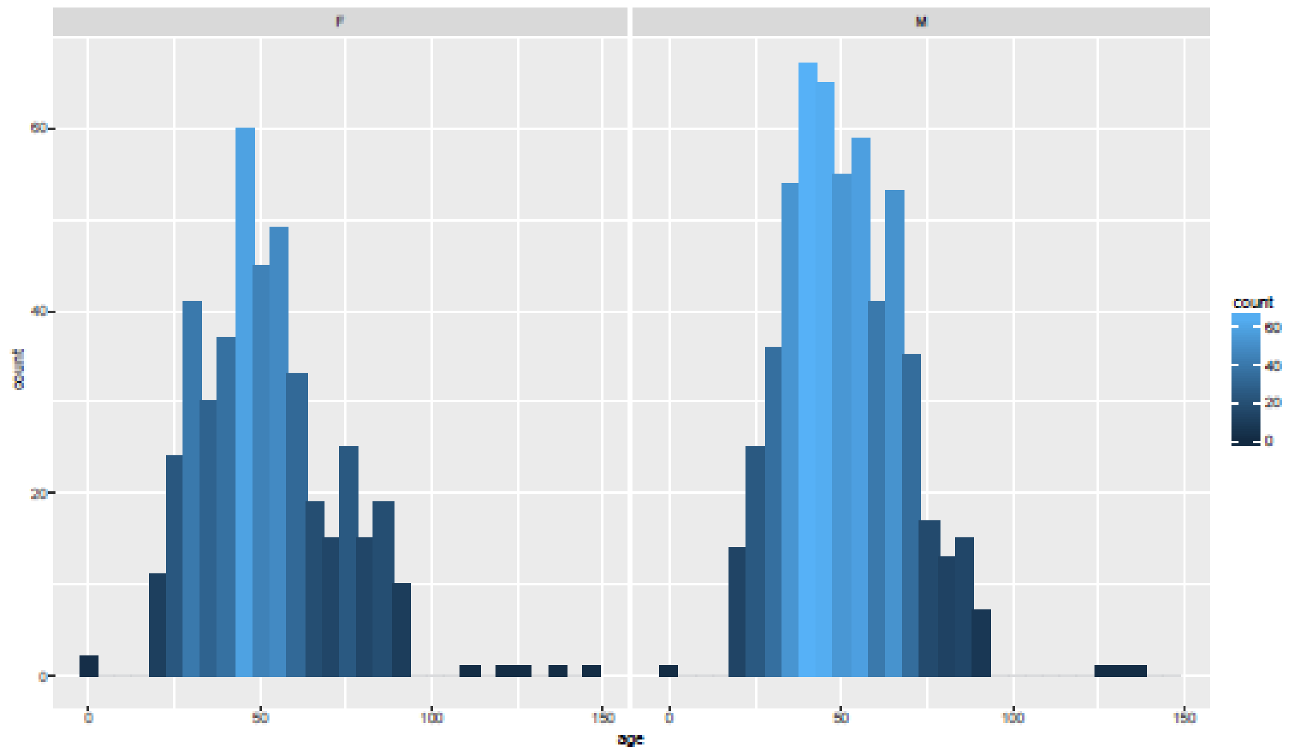
Boxplots in *ggplot*

```
> ggplot(data=census, aes(y=age, x=sex)) +
+   geom_boxplot()
```



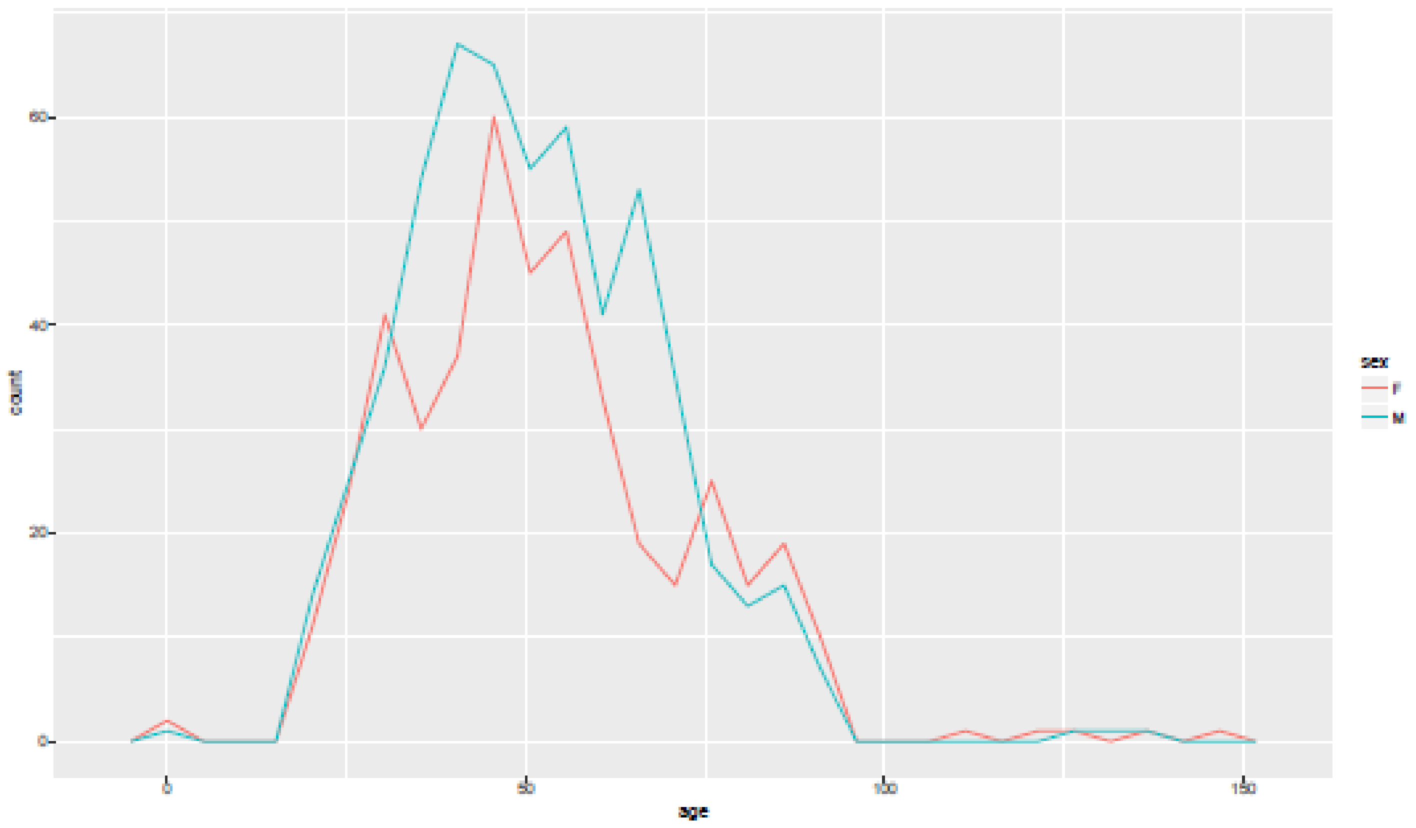
Facets in *ggplot*

```
> ggplot(data=census, aes(x=age)) +
+   geom_histogram(aes(fill=..count..)) +
+   facet_wrap(~sex)
```



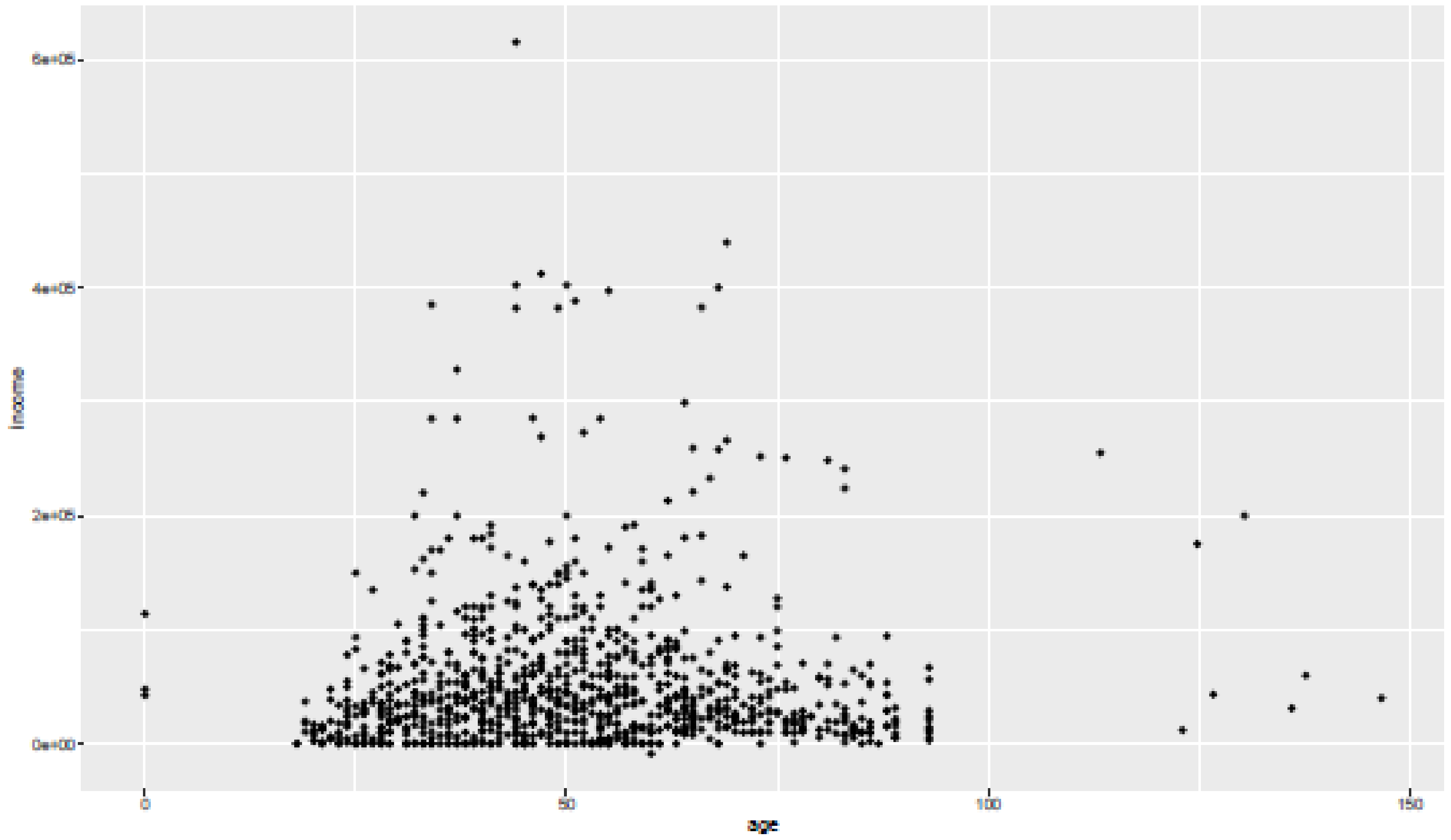
Frequency polygons in *ggplot*

```
> ggplot(data=census, aes(x=age)) +
+   geom_freqpoly(aes(color=sex))
```



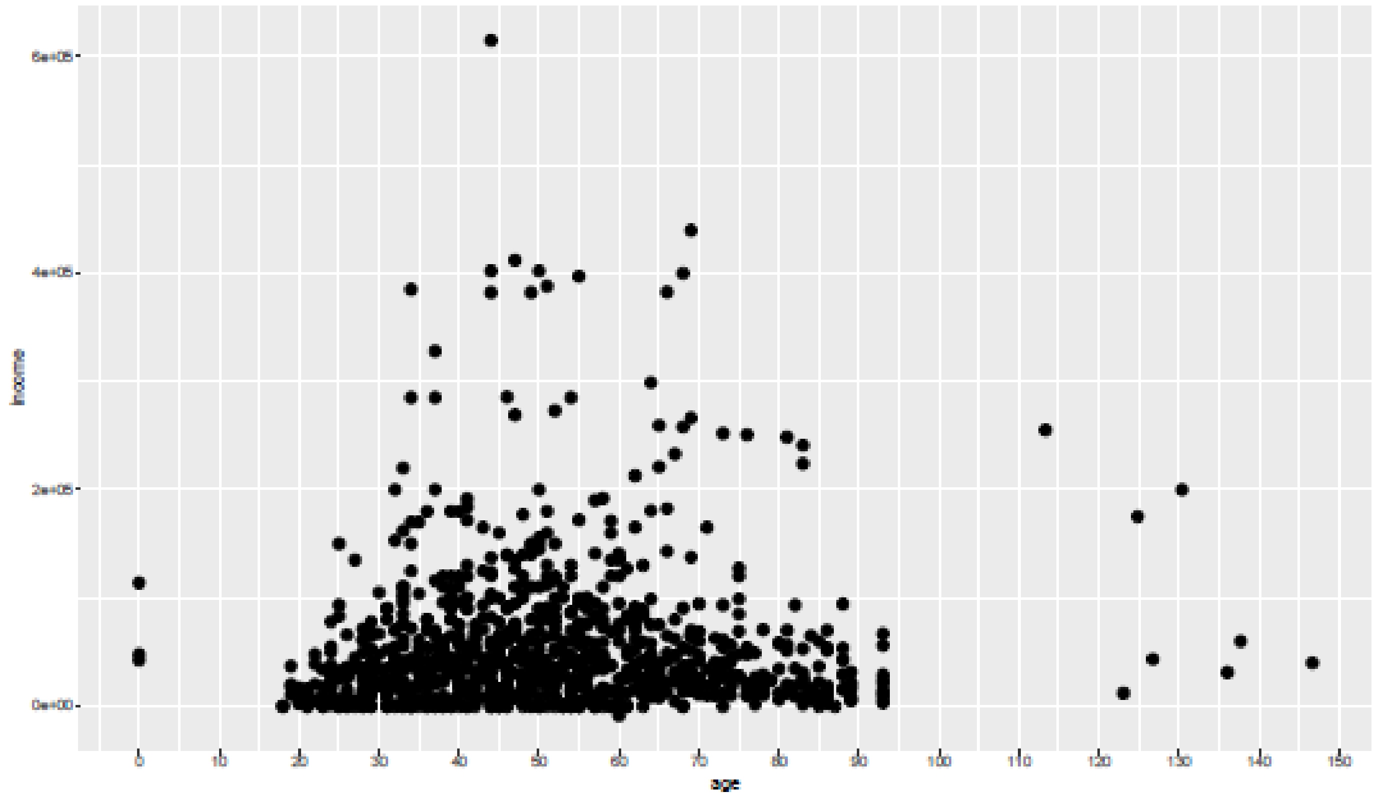
Scatterplots in *ggplot*

```
> ggplot(census, aes(age, income)) + geom_point()
```



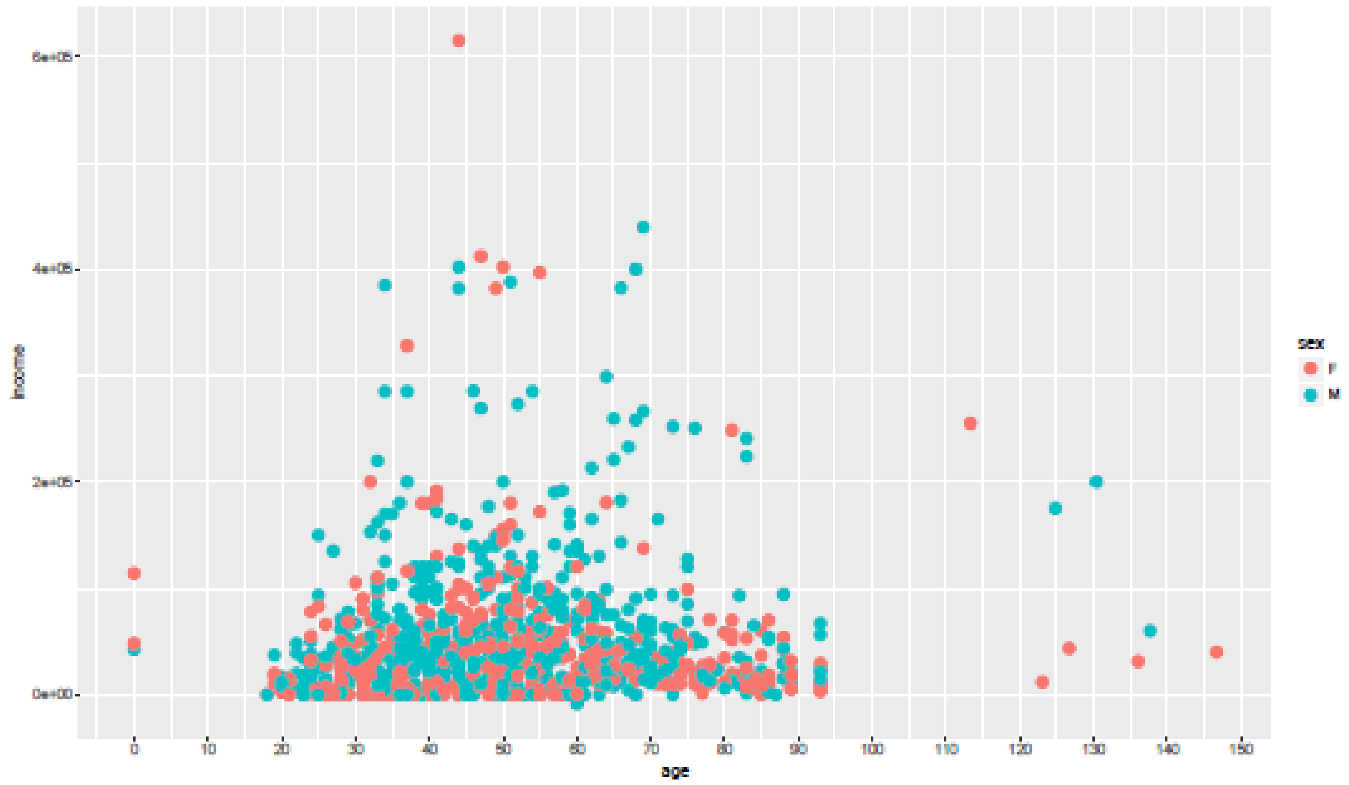
Scatterplots in *ggplot*

```
> ggplot(census, aes(age, income)) + geom_point(size=3) +
+   scale_x_continuous(breaks=seq(0,150,10))
```



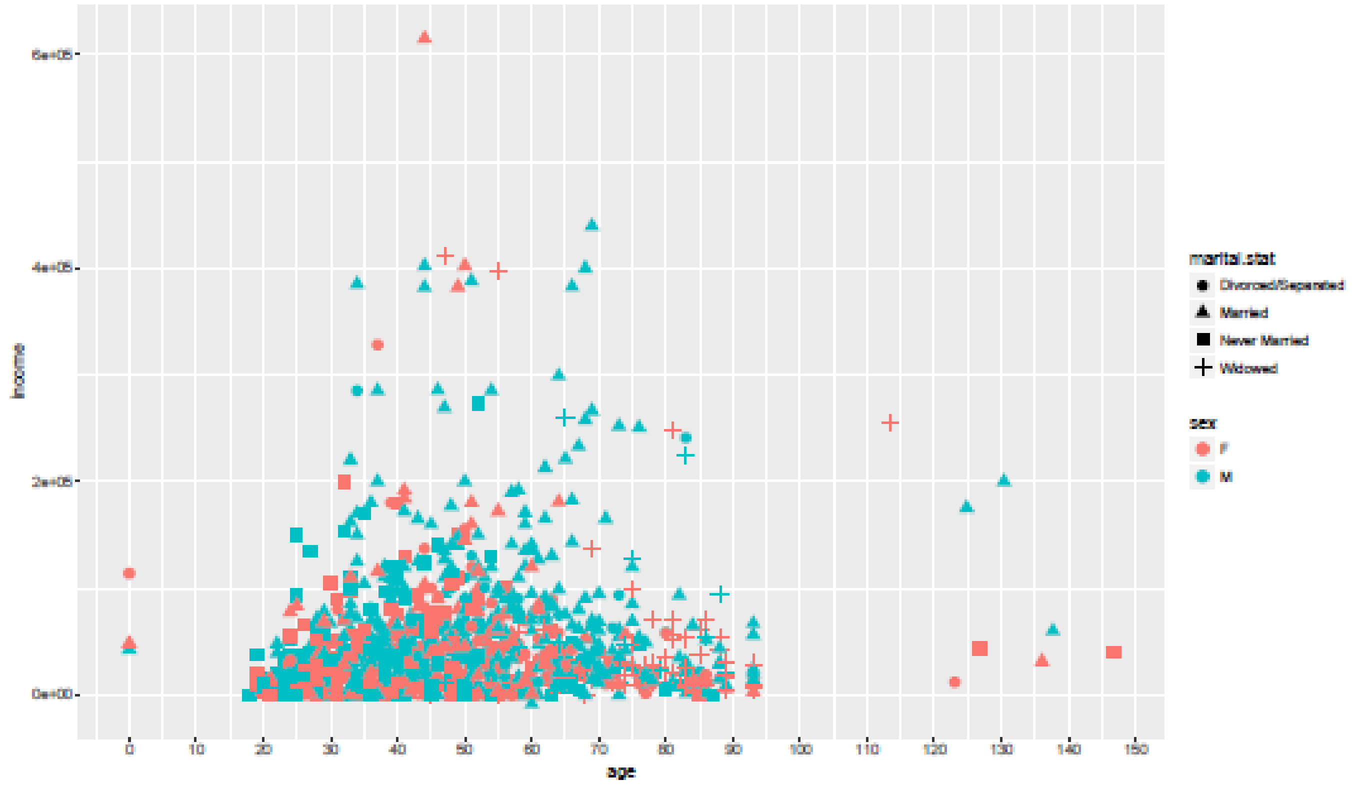
Scatterplots in *ggplot*

```
> ggplot(census, aes(age, income)) +
+   geom_point(aes(color=sex), size=3) +
+   scale_x_continuous(breaks=seq(0,150,10))
```



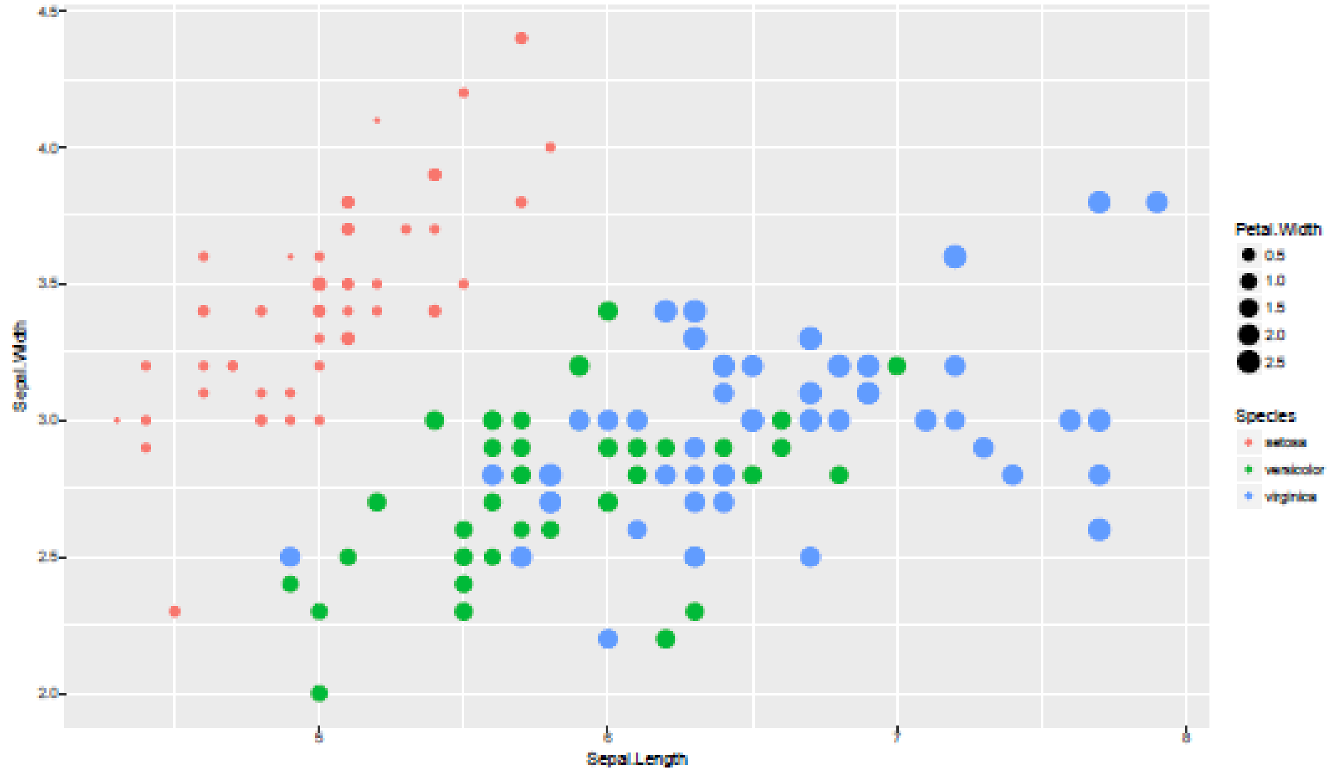
Scatterplots in *ggplot*

```
> ggplot(census, aes(age, income)) +
+   geom_point(aes(color=sex, shape=marital.stat), size=3)
+   scale_x_continuous(breaks=seq(0,150,10))
```



Scatterplots in *ggplot*

```
> ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
+   geom_point(aes(color=Species, size=Petal.Width))
```



Saving Plots as Images

- Plots can be saved in a variety of formats using functions with names that correspond to the extensions of the formats.
- For example the function pdf() saves a plot as a pdf.
- Functions for other formats include postscript(), bmp(), jpeg(), png() and tiff().

```
> jpeg(filename = "C:/r/scatterplot.jpg")
> hist(census$age)
> dev.off()
```