Basic Visualization

Wouter van Atteveldt

May 25, 2016

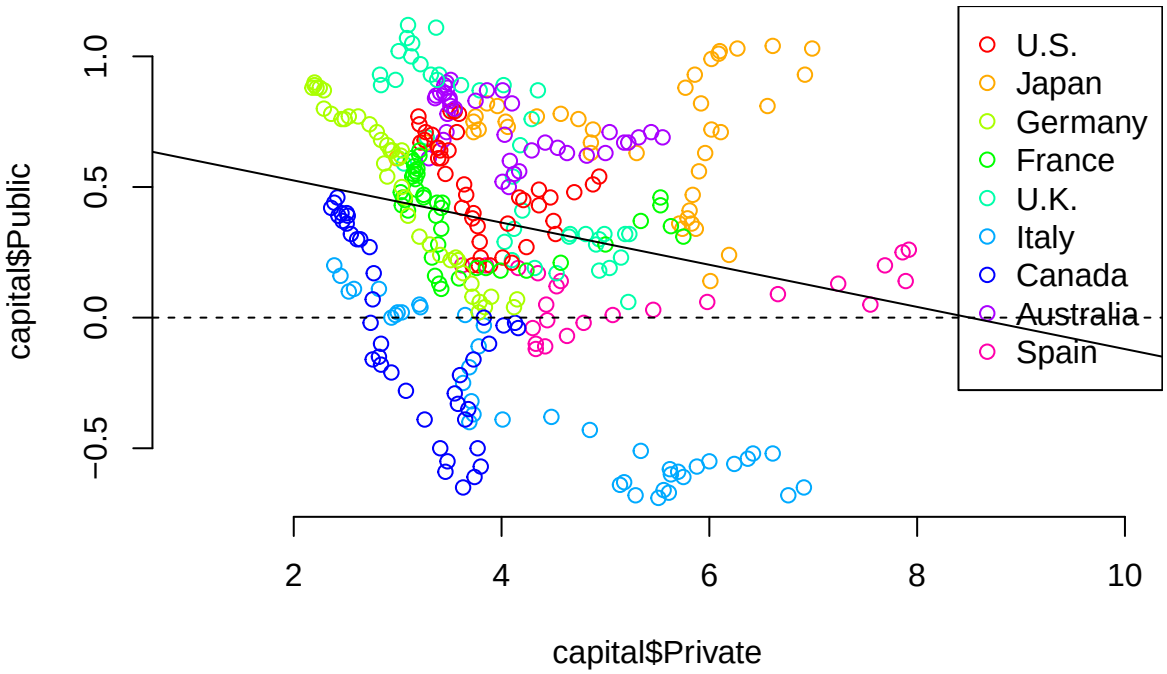
Plotting with base plot function

```
load("data/capital.rdata")
capital = na.omit(capital)
head(capital)
```

```
##   Year   Country Public Private Total
## 1 1970 Australia   0.61    3.30  3.91
## 2 1970    Canada   0.37    2.47  2.84
## 3 1970    France   0.41    3.10  3.51
## 4 1970   Germany   0.88    2.25  3.13
## 5 1970     Italy   0.20    2.39  2.59
## 6 1970     Japan   0.61    2.99  3.60
```

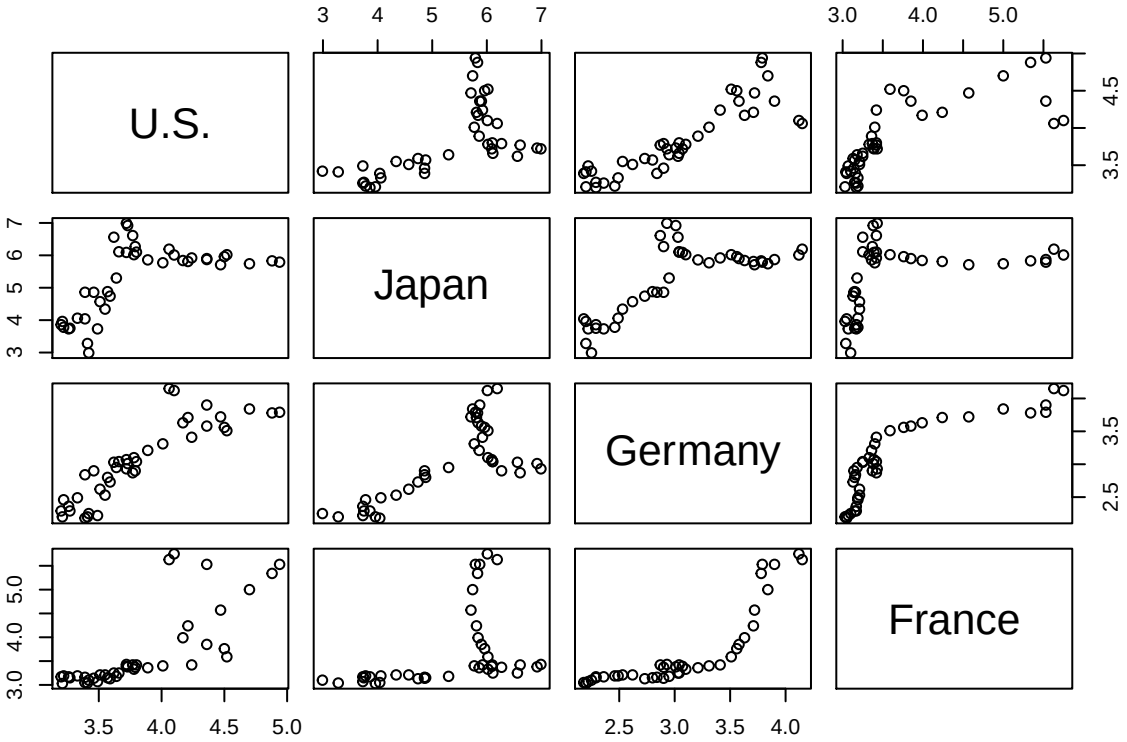
We can use `plot` to make a scatter plot of public to private capital, using colour to indicate country. We can also use abline to add a straight line (at y=0) and a linear fit to the plot:

```
col = rainbow(length(levels(capital$Country)))
plot(capital$Private, capital$Public, col=col[as.numeric(capital$Country)], frame.plot = F, xlim=c(1,10)
legend("topright", legend=levels(capital$Country), col = col, pch = 1)
abline(h = 0, lty=2)
abline(lm(capital$Public ~ capital$Private))
```



To see scatter plots of all country x country values for e.g. private capital, we first cast the data to wide format (to get the countries side-by-side) and then plot the data frame, taking only the columns we need:

```
library(reshape2)
wide = dcast(capital, Year ~ Country, value.var="Private")
plot(wide[2:5])
```



Histograms

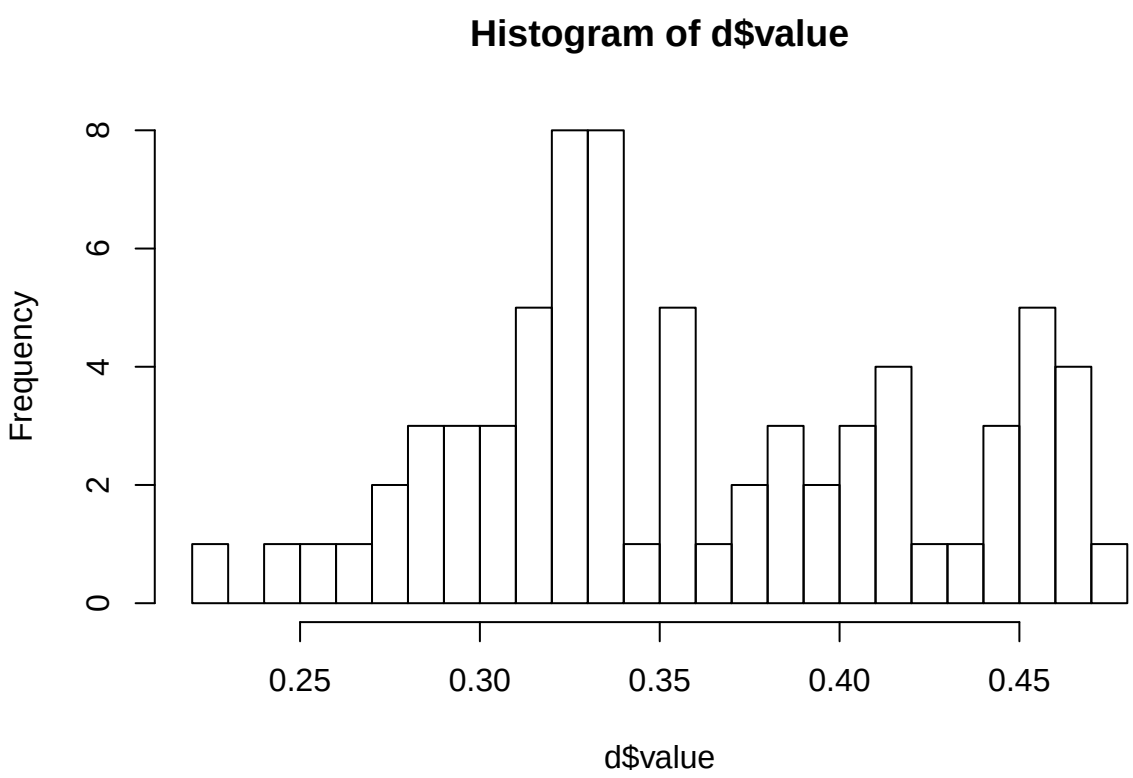
We can also take a histogram of the distrubution of income:

```
d = na.omit(read.csv("data/income_topdecile.csv"))
d = melt(d, id.var="Year")
head(d)
```

```
##   Year variable value
## 1 1900     U.S.  0.41
## 2 1910     U.S.  0.41
## 3 1920     U.S.  0.45
## 4 1930     U.S.  0.45
## 5 1940     U.S.  0.36
## 6 1950     U.S.  0.34
```

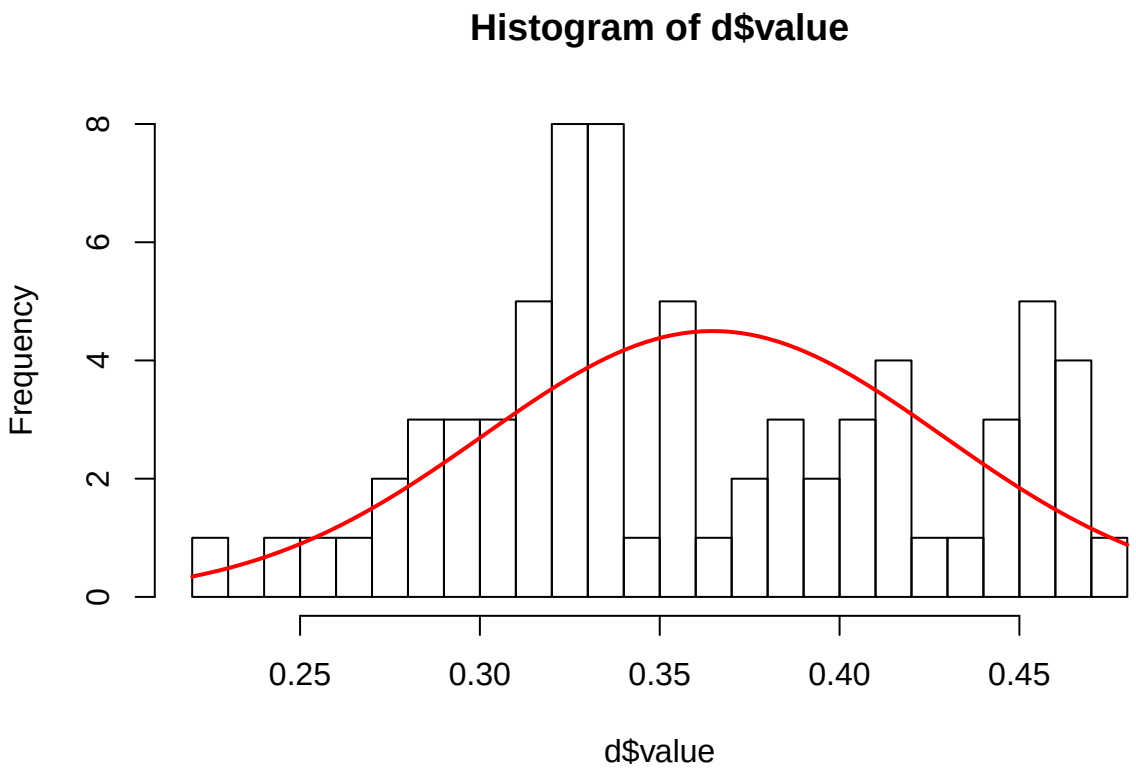
Plot the histogram using hist, specifying the amount of bins with `breaks`:

```
hist(d$value, breaks = 20)
```



You can also add a simulated normal value by using the `dnorm` function to calculate density based on mean and sd:

```
h = hist(d$value, breaks = 20)
x = seq(min(d$value), max(d$value), length.out=100)
norm = dnorm(x, mean(d$value), sd(d$value))
scale = max(h$counts / h$density, na.rm=T)
lines(x, norm * scale, col="red", lwd=2)
```



So it seems that the value is not normally distributed, which is confirmed by a Shapiro-Wilk text:

```
shapiro.test(d$value)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  d$value
## W = 0.95625, p-value = 0.01365
```

Plotting with ggplot2

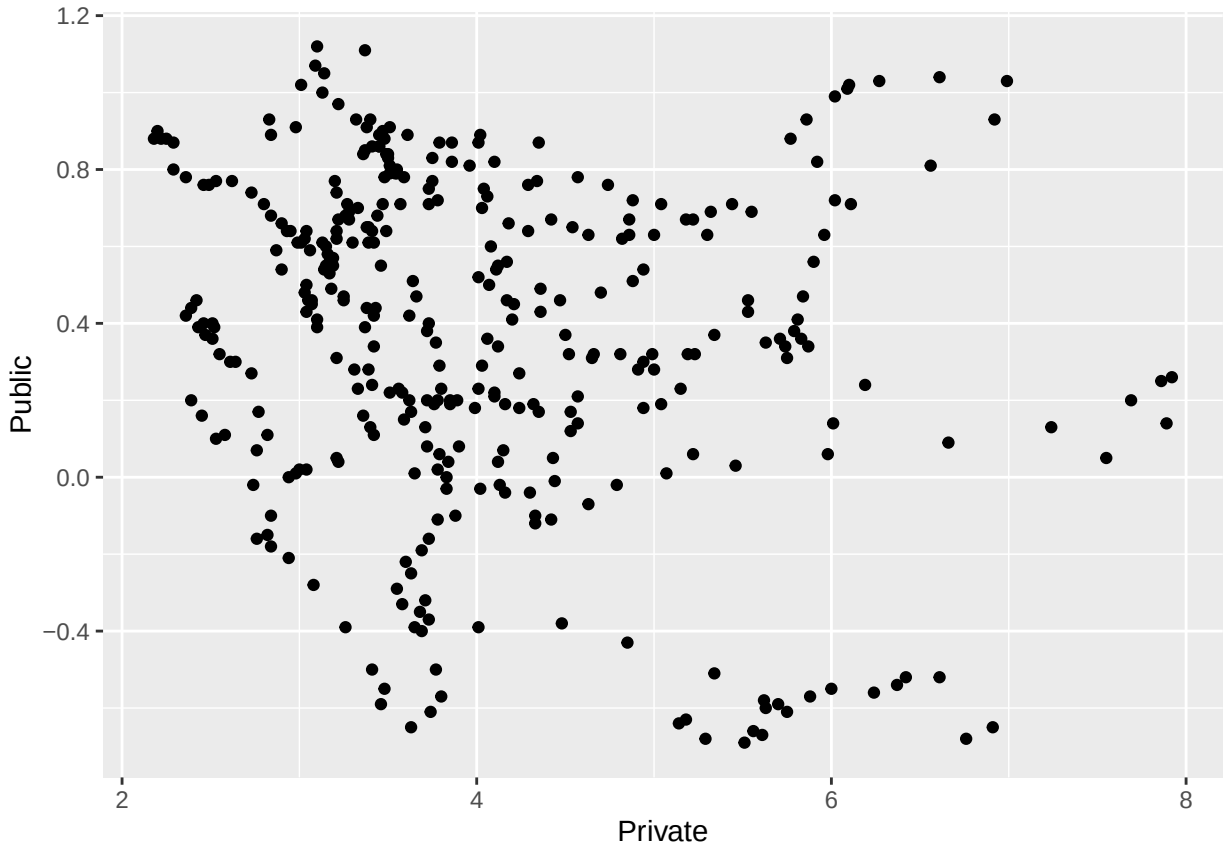
`ggplot2` is an advanced package for making graphs of all kinds. See http://www.r-graph-gallery.com/portfolio/ggplot2-package/ for a gallery of plots with associated R code.

In ggplot, a graph is composed of layers, which have an aesthetic mapping of data to visual properties, and geometries (lines, points, etc).

Scatter plots and regression lines

As a simple example, lets reproduce the scatter plot made above:

```
library(ggplot2)
ggplot(capital, aes(x=Private, y=Public)) + geom_point()
```



If we want to add a regresssion line, that would be another layer, in this case a line. Note that we need to change the mapping since y should now point to the fitted values of the model. We don't need to specify data and x again as these are taken from the base ggplot.

```
m = lm(Public ~ Private, data=capital)
regline =  geom_line(mapping=aes(y=fitted(m)))
ggplot(capital, aes(x=Private, y=Public)) + geom_point() + regline
```

(output not shown to save trees)

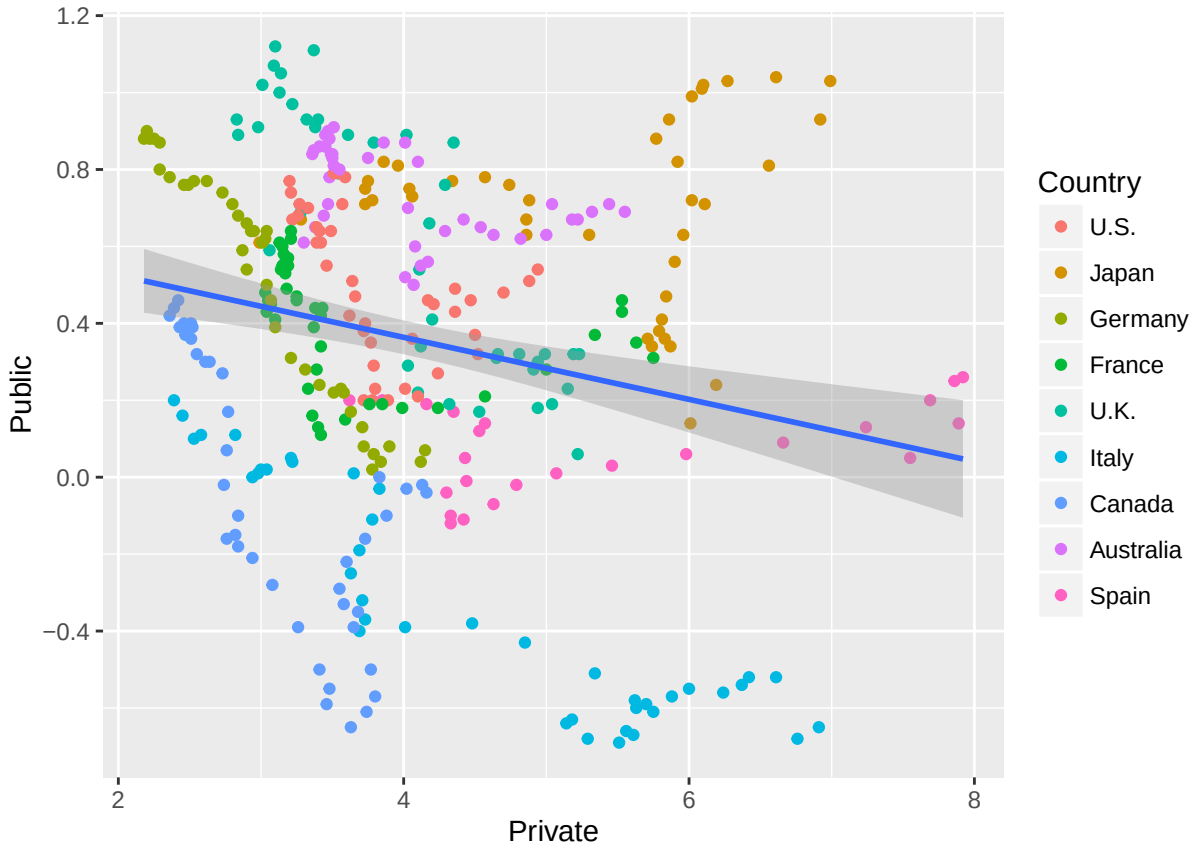
Now let's add a confidence interval band by using the predict function to get the interval. Note that we use `as.data.frame` to convert the predict output (a matrix) to a data.frame so the `$` works.

```
fit = as.data.frame(predict(m, interval = "confidence"))
band = geom_ribbon(mapping=aes(ymin=fit$lwr, ymax=fit$upr), alpha=.3)
ggplot(capital, aes(x=Private, y=Public)) + geom_point() + regline + band
```

(output not shown to save trees)

This can be done even more easily by using the `smooth` geom, which automatically adds a regression line and confidence interval. Moreover, we can use the `color` aesthetics to get points color per country:

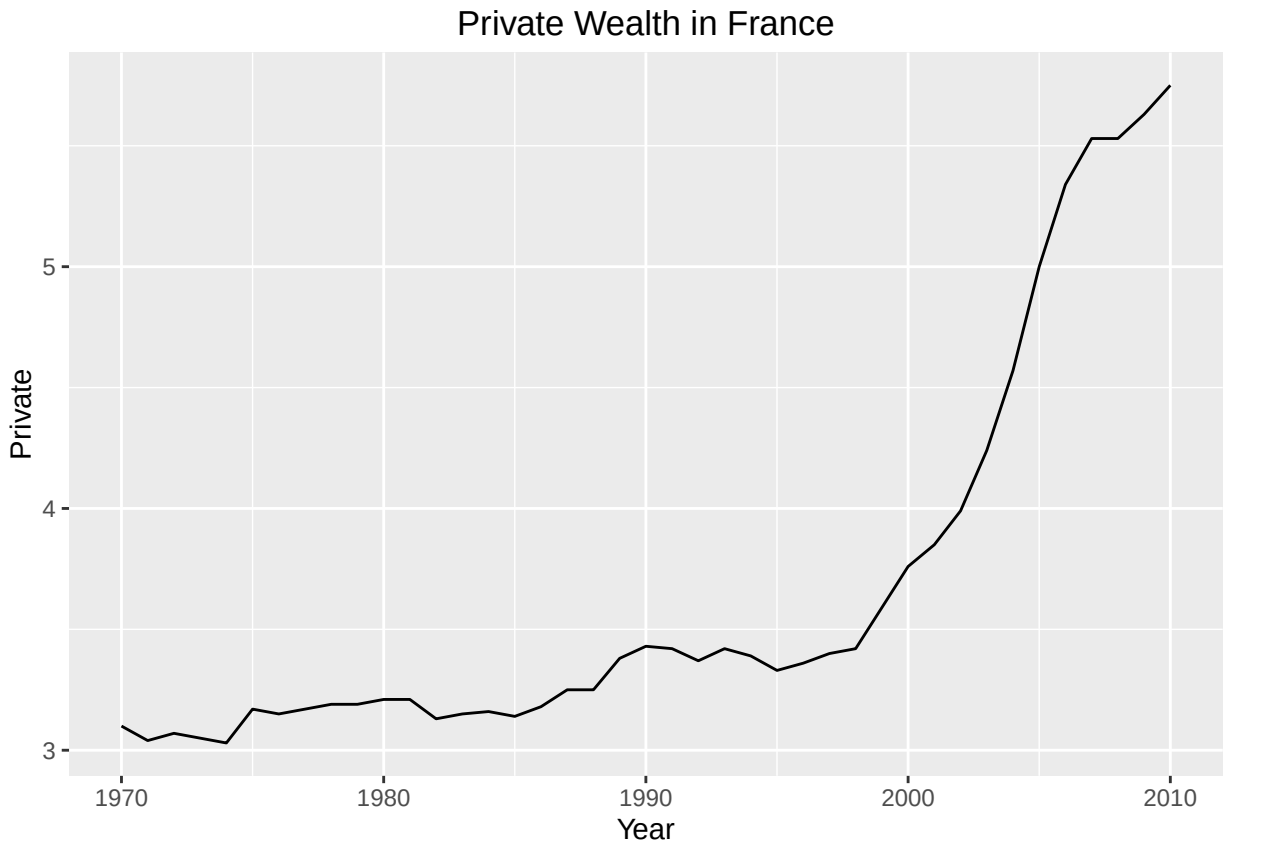
```
ggplot(capital, aes(x=Private, y=Public)) +
  geom_point(mapping=aes( color=Country)) + geom_smooth(method='lm')
```


1.2
0.8
0.4
0.0
−0.4
Public
2
4
6
8
Private
Country
U.S.
Japan
Germany
France
U.K.
Italy
Canada
Australia
Spain

Line plots

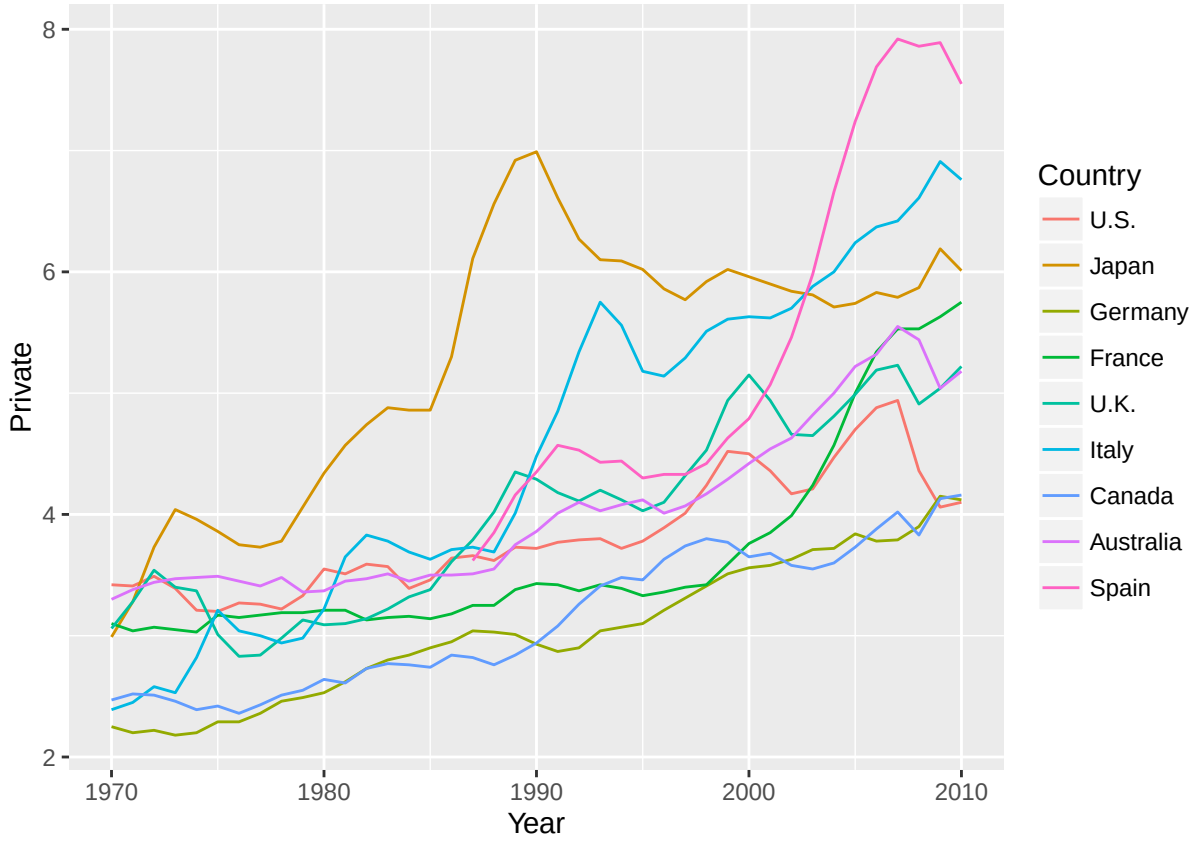
We can ues the geom_line to create a simple line plot, e.g. for the development of French private wealth:

```
d = subset(capital, Country=="France")
ggplot(d, aes(x=Year, y=Private)) + geom_line() + ggtitle("Private Wealth in France")
```



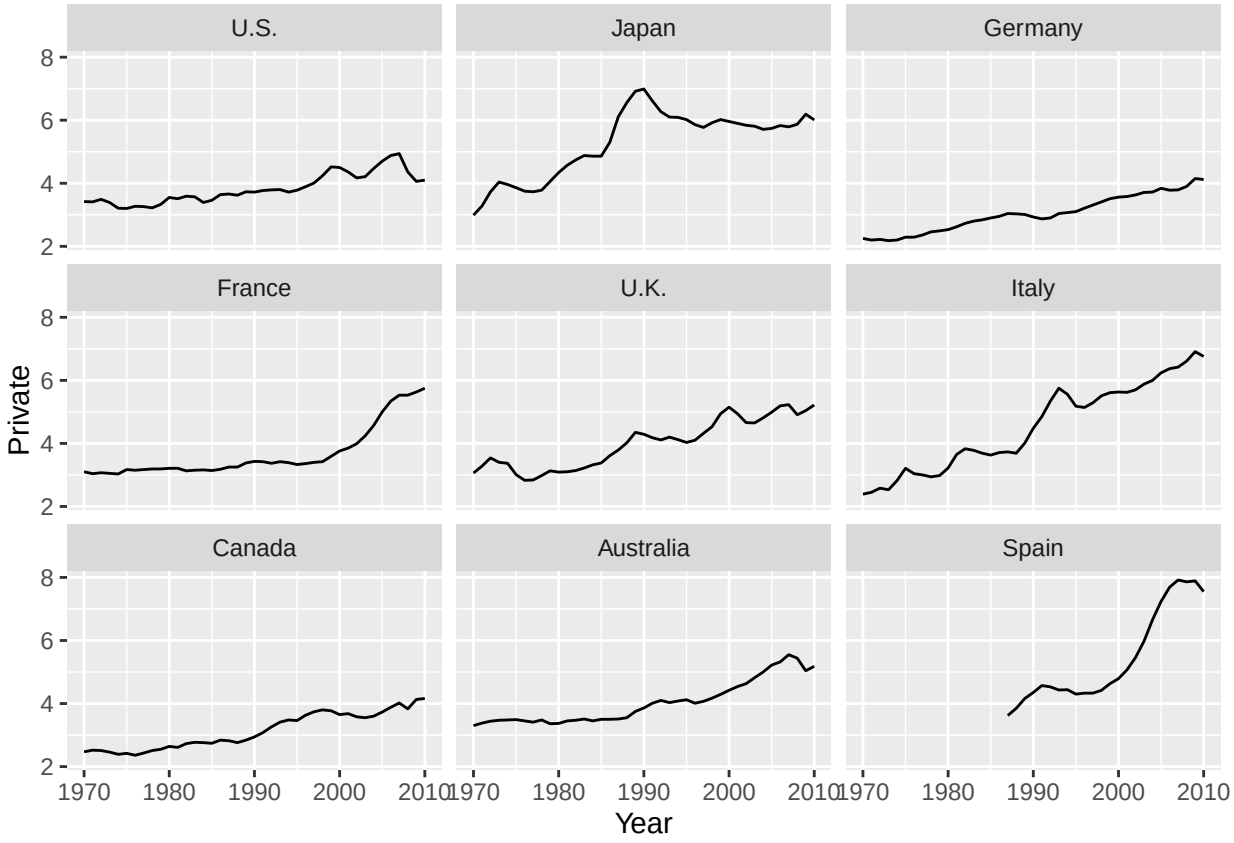
You can make a line plot with multiple lines by specifying the color mapping:

```
ggplot(capital, aes(x=Year, y=Private, color=Country)) + geom_line()
```



Finally, we can use 'faceting' to automatically create a plot per country:

```
ggplot(capital, aes(x=Year, y=Private)) + geom_line() + facet_wrap(~Country)
```



GoogleVis

Google has an online API for creating interactive graphs that can be included in HTML reports/presentations or on webpages.

```
library(googleVis)
wide = dcast(capital, Year ~ Country, value.var = "Private")
plot(gvisLineChart(wide, xvar = "Year", yvar = colnames(wide)[-1]))
```

(run code interactively to see results)