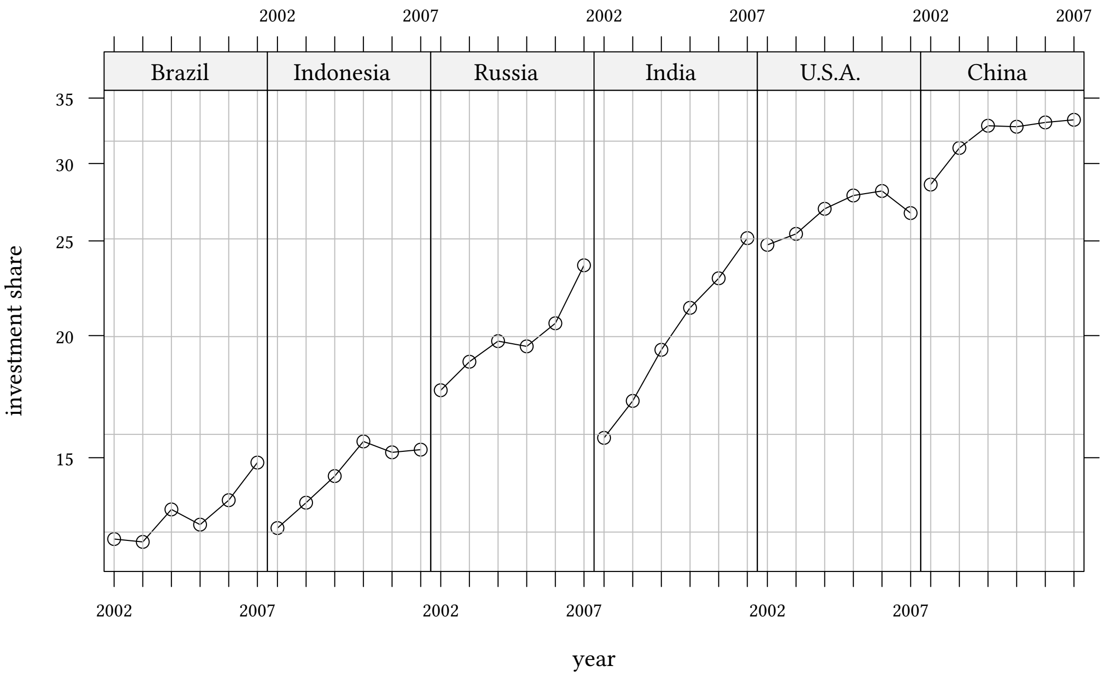
Using Graphs and Visualising Data



Oliver Kirchkamp

Contents

1 Introduction

1.1 Literature

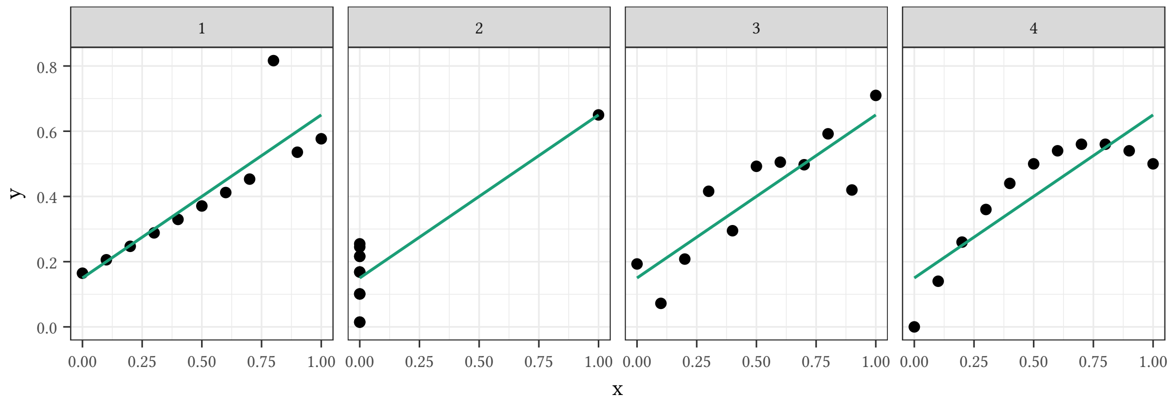
- The Elements of Graphing Data (Revised Edition). W. S. Cleveland (1994). Hobart Press, Summit, New Jersey, U.S.A.
- The Visual Display of Quantitative Information. Edward Tufte (2001). Bertrams.

1.2 Examples

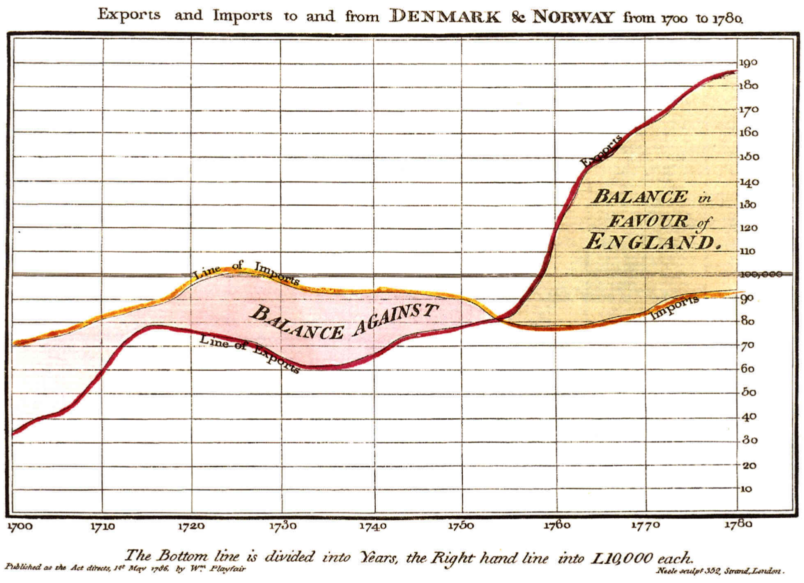
The following four datasets all have the same correlation $\rho = 0.8730379$.

They all have the same regression coefficients β_0=0.15 and β_1=0.5.

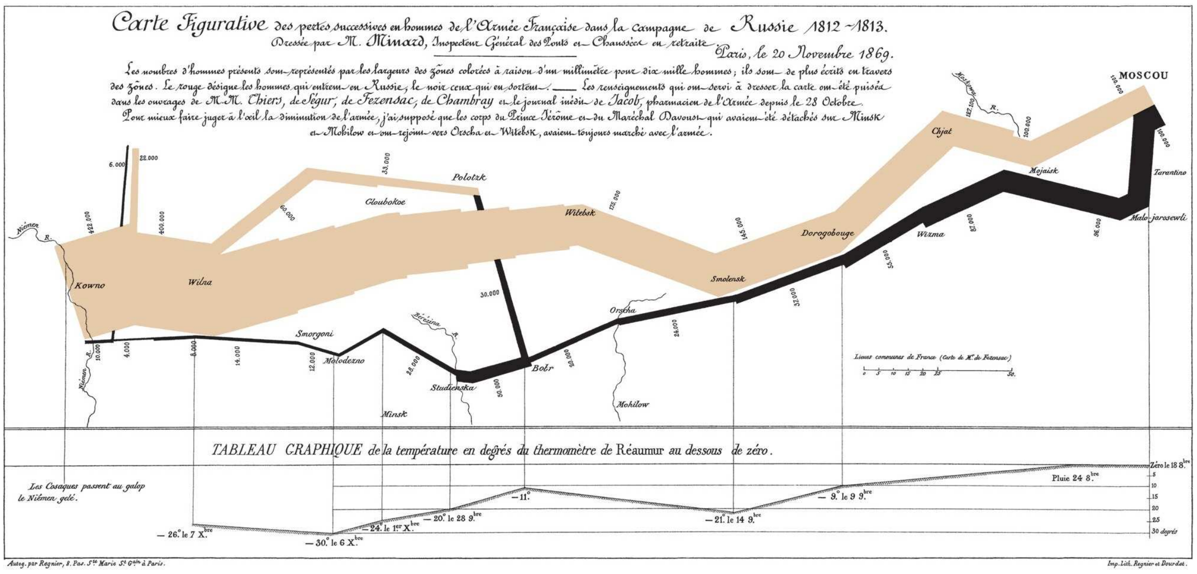
x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
0.00	0.16	0.00	0.17	0.00	0.19	0.00	0.00
0.10	0.21	0.00	0.01	0.10	0.07	0.10	0.14
0.20	0.25	0.00	0.10	0.20	0.21	0.20	0.26
0.30	0.29	0.00	0.25	0.30	0.42	0.30	0.36
0.40	0.33	0.00	0.10	0.40	0.29	0.40	0.44
0.50	0.37	0.00	0.25	0.50	0.49	0.50	0.50
0.60	0.41	0.00	0.22	0.60	0.51	0.60	0.54
0.70	0.45	0.00	0.17	0.70	0.50	0.70	0.56
0.80	0.82	0.00	0.22	0.80	0.59	0.80	0.56
0.90	0.54	0.00	0.01	0.90	0.42	0.90	0.54
1.00	0.58	1.00	0.65	1.00	0.71	1.00	0.50



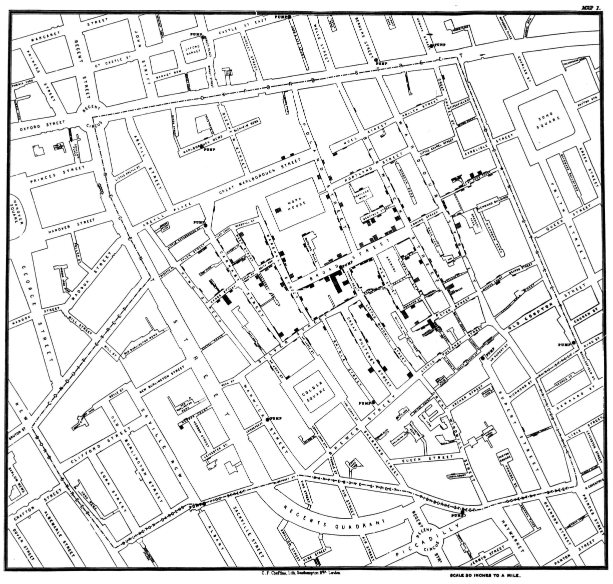
William Playfair's trade-balance time-series chart, 1786:



Charles Minard's 1869 chart of Napoleon's 1812 Russian campaign:

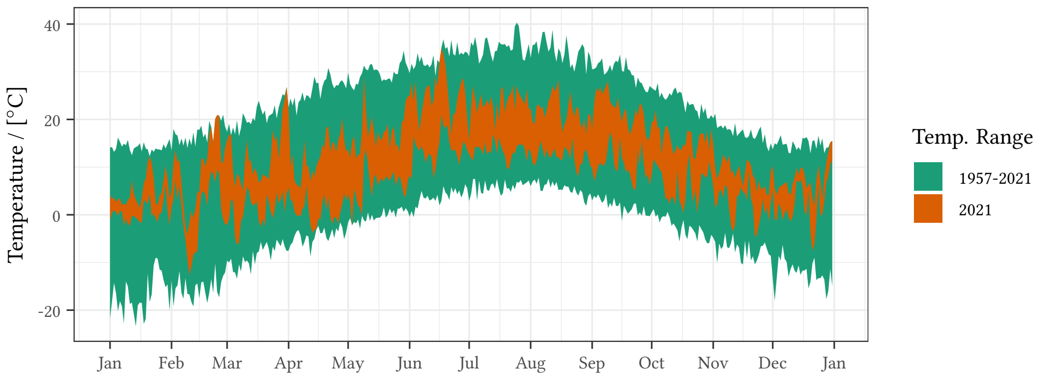


John Snow, 1854 Broad Street cholera outbreak:

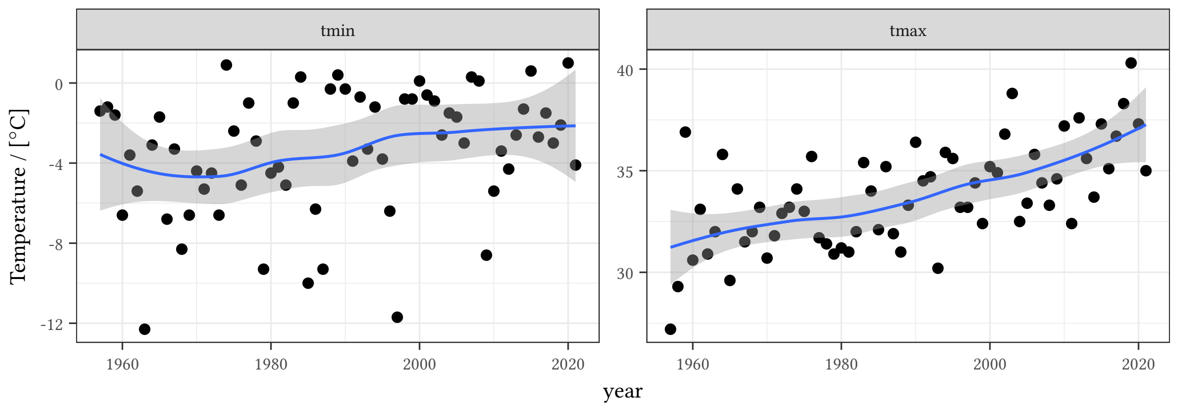


Data to Ink ratio — Daily Max/Min-Temperature in Erfurt/Weimar 1957 – 2021

46996 numbers:



Climate change in Köln/Bonn from 1951-2021?



Aims:

- Note: good graphs are self-explanatory!
  → The key to understand a graphs should not be hidden somewhere in the text!
- Often, the optimal presentation of data is not "standard".
- There are no "recipes" how to present data.
- We have to use our own imagination.
- Still, some examples might help.

What can be achieved with a good graph?

A graph can...

- ...make the reader familiar with the structure of the data (create trust),
- ...motivate a research question,
- ...summarise conclusions of the paper.

A graph must be excellent:

- Some readers look only or mainly at the figures and the graphs.
- Each graph should tell a story and should be self-explanatory.
- Among the many ways to present our data and our results, we have to chose the best way.

Presenting only aggregate statistics, tests and estimates might discard relevant structure:

1.3 Properties of good graphs

- Essential items are shown clearly.
- Superfluous items are not shown.
- All elements of the graph are explained in the figure (not only in the text).

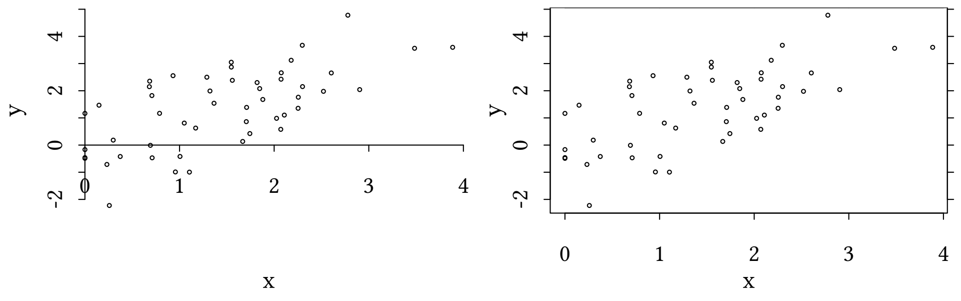
1.4 How to present

1.4.1 Axes

Frames: The following two graphs show the same data. However, in the graph on the left the axes and the data points are superimposed. In the graph on the right axes and data points are separate items. A frame around the plot region makes it more clear where the reader should expect data.

```
set.seed(131)
N<-50
x<-rnorm(n=N,mean=1.5)
y<-x+rnorm(N)
x<-pmax(0,x)
#
z<-rbinom(N,2,.5)
Treatment <- factor(z)
labels<-c("Baseline","A","B")
levels(Treatment)<-labels
gData <- data.frame(x,y,z,Treatment)
```

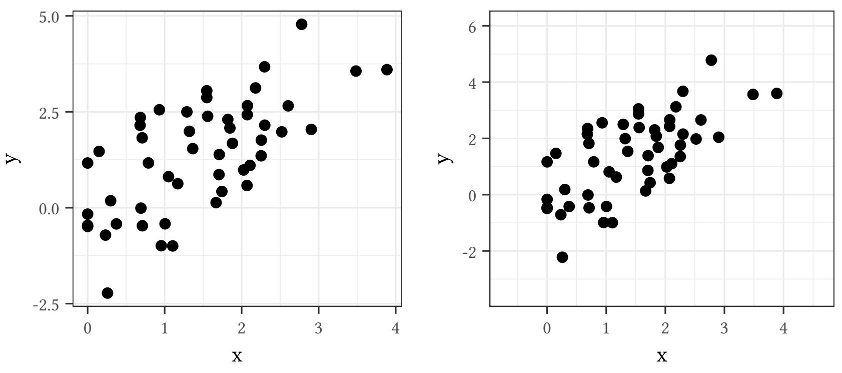
```
par(mfrow=c(1,2),mar=c(4,3,0,1),mex=.5)
plot(y ~ x,axes=FALSE,cex=.25)
axis(side=1,pos=0)
axis(side=2,pos=0)
plot(y ~ x,cex=.25)
axis(3,labels=FALSE)
axis(4,labels=FALSE)
```



- Labels and tick marks are separated from the data.
- Ticks on the opposite axis can help.

Ranges: In the following example both graphs show the same data. The only difference is the range of the axes.

```
p1 <- ggplot() + geom_point(aes(x,y))
p2 <- ggplot() + geom_point(aes(x,y)) +
      scale_x_continuous(expand=expansion(mult=.25)) +
      scale_y_continuous(expand=expansion(mult=.25))
grid.arrange(p1,p2,nrow=1)
```



Ranges are chosen such that...

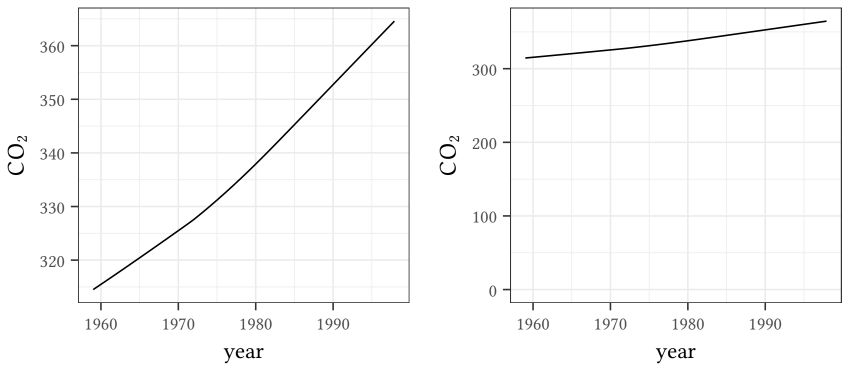
- ...all data is included,
- ...space is used in an efficient way.

Ranges and correlations We tend to perceive more correlation if the data occupies less space.

- The amount of white space around the data should be similar in all graphs.

Ranges that include zeroes: Mauna Loa Atmospheric CO_2 Concentration:

```
co2 |>
    lowess() |>
    data.frame() |>
    ggplot() + geom_line(aes(x,y)) +
    labs(y="$CO_2$",x="year") -> p1
p1 + expand_limits(y=0) -> p2
grid.arrange(p1,p2,nrow=1)
```

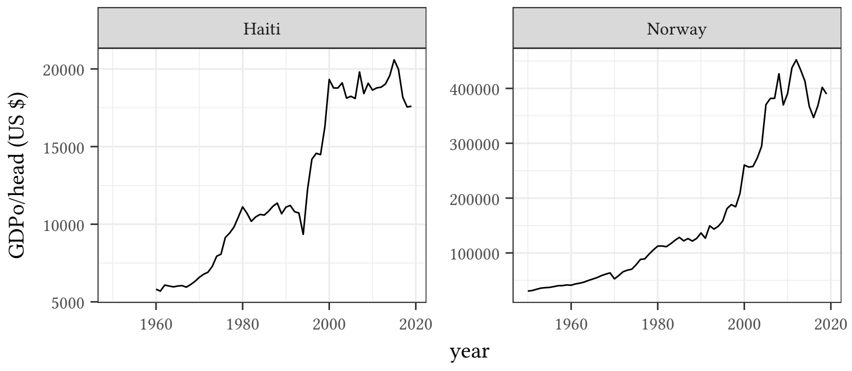


It can be helpful to include zero, but it can also waste space.

Comparable scales: In the following example we use different scales for the two diagrams. This makes them difficult to compare (although space is used efficiently).[1]

```
library(pwt10)
data(pwt10.0)
pwt10.0 %>%
    filter(country %in% c("Norway","Haiti")) %>%
    select(c("cgdpo","pop","country","year")) %>%
    mutate(gdp = cgdpo/pop) -> pwt
```

```
ggplot(pwt) + geom_line(aes(x=year,y=cgdpo)) +
    labs(y="GDPo/head (US \\$)") +
    facet_wrap(vars(country),scales="free")
```



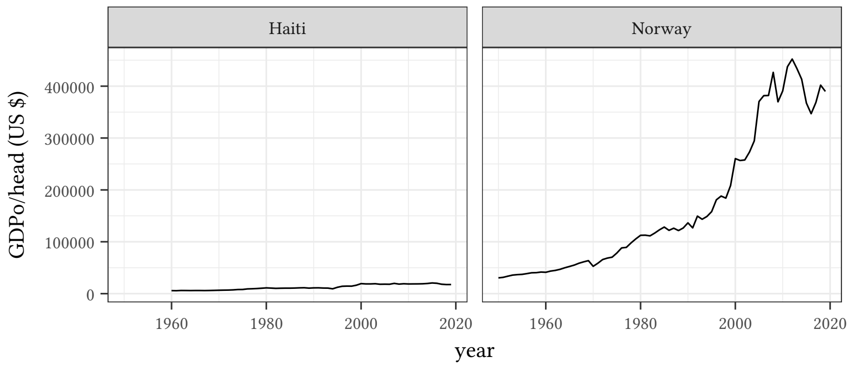
The figure shows real gross domestic product (GDPo) per capita (US dollars in 2017 prices).

Data is taken from Penn World Table Version 10.0.

In the next figure we use the same scale for the two diagrams. Now we see immediately that GDPo is larger in Brazil and smaller in Indonesia. Of course, presenting both lines in one diagram might be preferable, here.

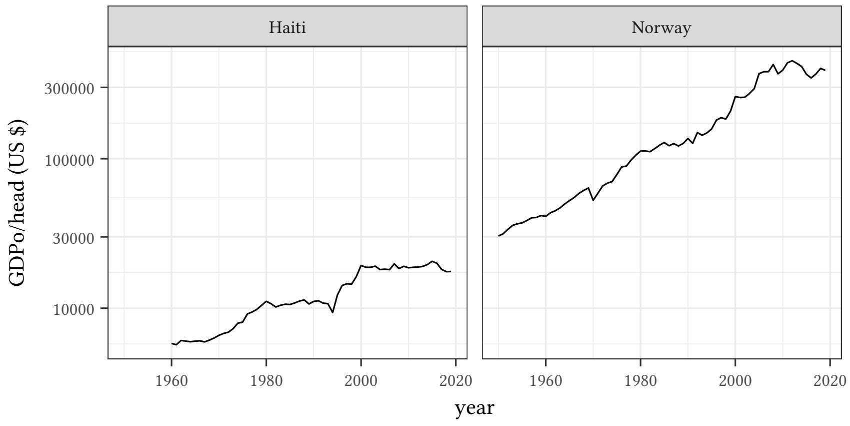
[1]Data from Alan Heston, Robert Summers and Bettina Aten, Penn World Table Version 10.0, Center for International Comparisons of Production, Income and Prices at the University of Pennsylvania, August 2009.

```
ggplot(pwt) + geom_line(aes(x=year,y=cgdpo)) +
    labs(y="GDPo/head (US \\$)") +
    facet_wrap(vars(country),scales="fixed")
```



The figure shows real gross domestic product (GDPo) per capita (US dollars in 2017 prices). Data is taken from Penn World Table Version 10.0.

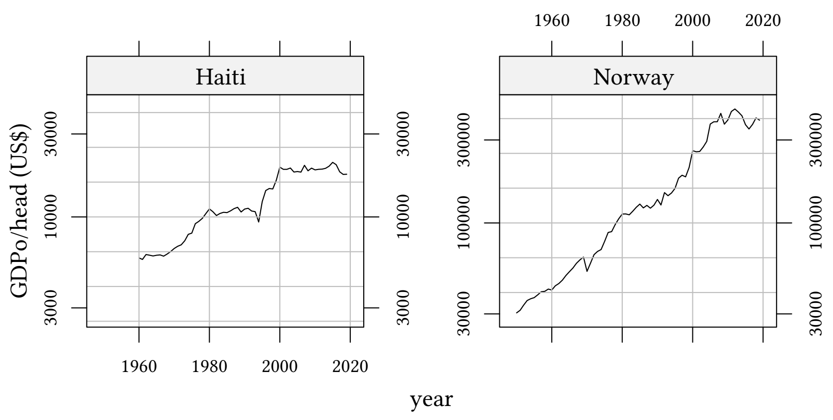
```
ggplot(pwt) + geom_line(aes(year,y=cgdpo)) +
    labs(y="GDPo/head (US \\$)") +
    facet_wrap(vars(country),scales="fixed") +
    scale_y_log10()
```



A logarithmic scale facilitates comparing relative growth. Also with logs, comparability does not require to use the *same axes* for several diagrams. Sometimes it might be better to use the *same scale* with a different origin.

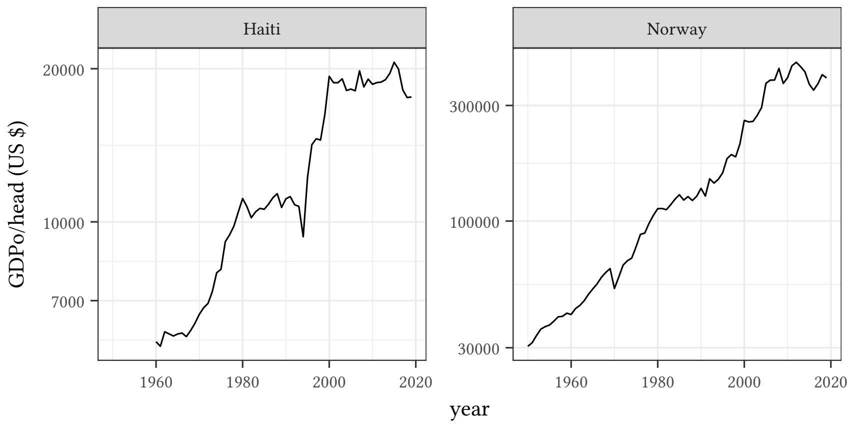
Another possibility are sliced scales, i.e. the same scale, but different origins:

```
xyplot(cgdpo ~ year | country,data=pwt,type="l",
       scales=list(y=list(log=10,relation="sliced")),
       between=list(x=2),
       par.settings = list(layout.widths = list(right.padding = 5)),
       yscale.components=yscale.components.log10.3,
       ylab="GDPo/head (US\\$)")
```



A sliced logarithmic scale facilitates comparing relative growth even more.

```r
ggplot(pwt) + geom_line(aes(year,y=cgdpo)) +
    labs(y="GDPo/head (US \\$)") +
    facet_wrap(vars(country),scales="free") +
    scale_y_log10()
```

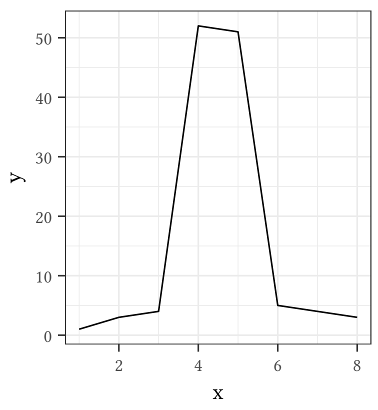


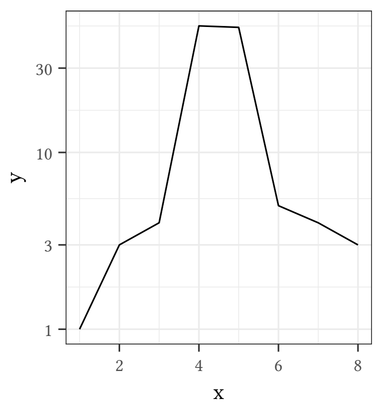
The figure shows real gross domestic product (GDPo) per capita (dollars in 2017 prices). Data is taken from Penn World Table Version 10.0.

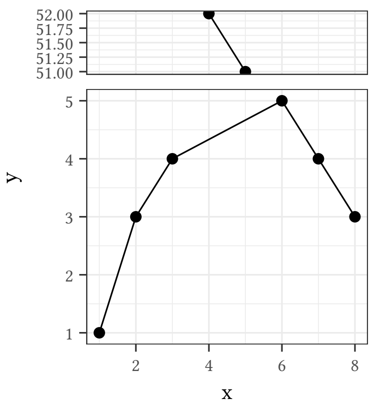
Breaks:

```r
data.frame(x=1:8,y=c(1,3,4,52,51,5,4,3)) %>%
    mutate(out=y > mean(c(max(y),min(y))),
           shrink=min(y[out])-max(y[!out])-2,
           yShrink=y - ifelse(out,shrink,0)) -> dBreak
```

```r
p1<-ggplot(dBreak,aes(x=x,y=y)) + geom_line()
p2<-ggplot(dBreak,aes(x=x,y=y)) + geom_line() + scale_y_log10()
p3<-ggplot(dBreak,aes(x=x,y=y)) + geom_line() + geom_point() +
    facet_grid(out ~ .,scale="free_y",space="free_y",as.table=FALSE) +
    theme(strip.text.y = element_blank())
grid.arrange(p1,p2,p3,nrow=1)
```







All three graphs try to show the same data. The first graph uses a linear scale. Here the outlier is clearly visible.

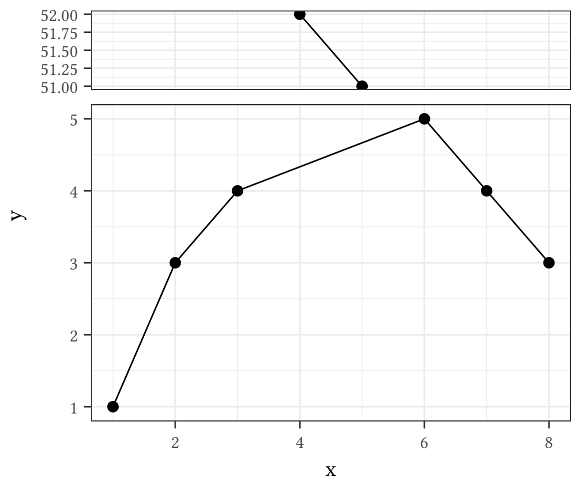
The graph in the middle uses a logarithmic scale. This can be reasonable if ratios of the variable are interesting.

The last graph tries to save space by "breaking" the axis. If gaps can not be avoided, dividing the graph into different panels might be preferable (the `shingle` function might help to find divisions).

`ggplot` can "slice" axes, however I find the result not entirely convincing:

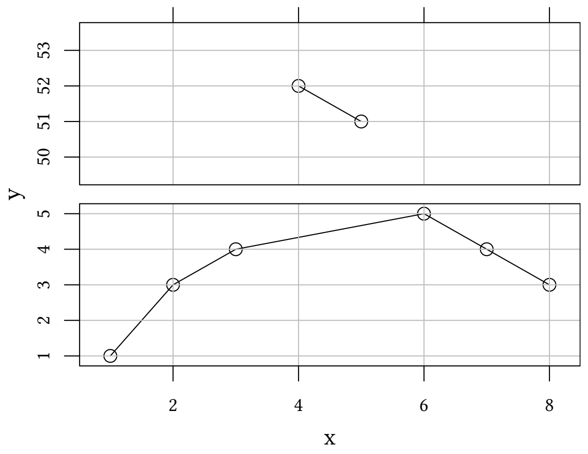
GGplot:

```r
ggplot(dBreak,aes(x=x,y=y)) + geom_line() +
    geom_point() +
    facet_grid(out ~ .,scale="free_y",
               space="free_y",as.table=FALSE) +
    theme(strip.text.y = element_blank())
```



Lattice:

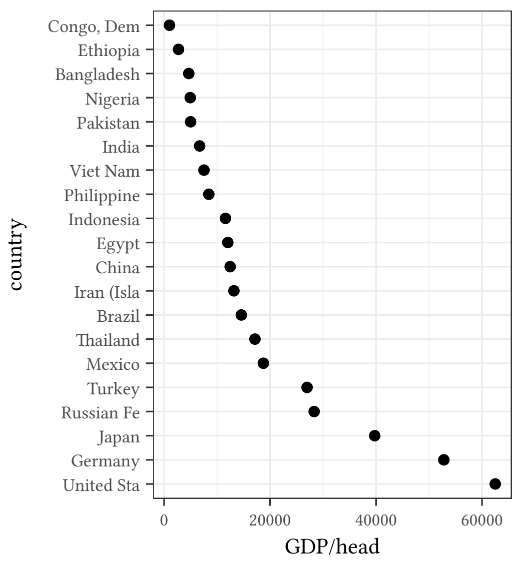
```
xyplot(y ~ x|out,data=dBreak,layout=c(1,2),
       strip=FALSE,scales=list(y="sliced"),
       between=list(y=.5),type="o")
```



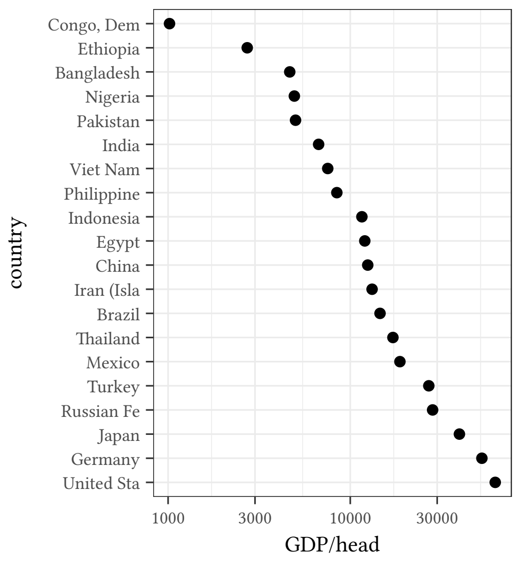
Logarithmic scales: The first graph shows GDPo on a linear scale, the second one uses a logarithmic scale. On a linear scale countries like Congo and Ethiopia seem to be quite similar, Germany and U.S.A. look distinct.

The log scale makes it easier to compare ratios. We see that, in relative terms, Germany is perhaps closer to the United States of America than Congo is to Ethiopia.

```
N<-20
pwt10.0 %>% filter(year==2019 & country!="China Version 2") %>%
    filter(pop >= -sort(-pop)[N]) %>%  ## only the top N countries
    mutate(country=substr(country,1,10)) %>%
    mutate(country=reorder(country,-cgdpo/pop),
           gdp=cgdpo/pop) -> pwtG20
ggplot(pwtG20,aes(x=gdp,y=country)) + geom_point() + labs(x="GDP/head")
```



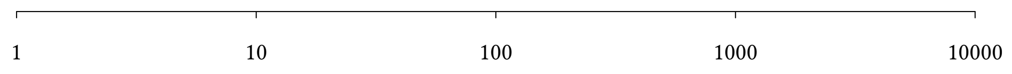
```
ggplot(pwtG20,aes(x=gdp,y=country)) + geom_point() + scale_x_log10() + labs(x="GDP/head")
```



Which logarithm?

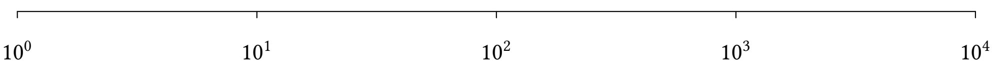
When the ratio of the largest to the smallest value is really large, then a logarithmic scale is easy to grasp.

```
par(mar=c(4,0,0,0),mex=.5)
plot(NULL,xlim=c(.5,20000),ylim=c(0,1),axes=FALSE,log="x",ylab="",xlab="")
axis(1)
```



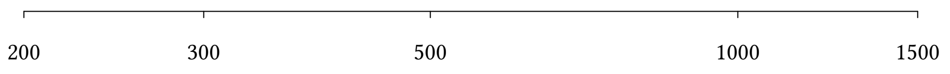
As an alternative we could also write the powers of 10:

```
labs<-10^(0:4)
plot(NULL,xlim=c(.5,20000),ylim=c(0,1),axes=FALSE,log="x",ylab="",xlab="")
axis(1,at=labs,labels=paste("$10^",log10(labs),"$"))
```



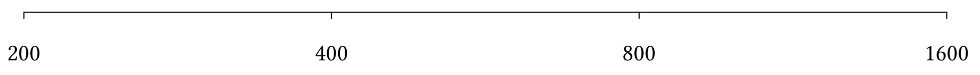
The scale is harder to grasp when the ratio is less extreme:

```
labs<-c(200,300,500,1000,1500)
plot(NULL,xlim=c(150,1800),ylim=c(0,1),axes=FALSE,log="x",ylab="",xlab="")
axis(1,at=labs,labels=labs)
```



Here using a logarithm with base two can help:

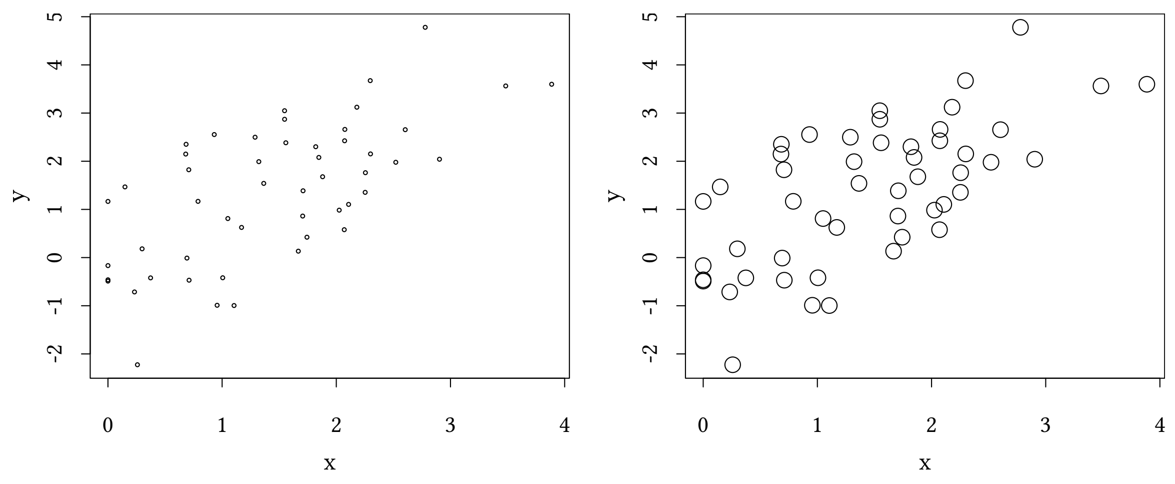
```
labs<-100*(2^(1:4))
plot(NULL,xlim=c(150,1800),ylim=c(0,1),axes=FALSE,log="x",ylab="",xlab="")
axis(1,at=labs,labels=labs)
```



1.4.2 Points

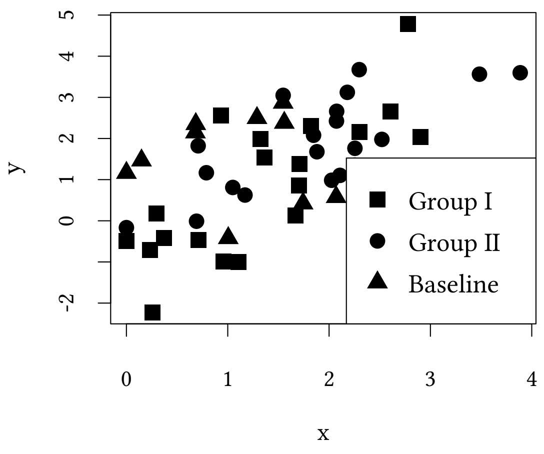
Size In the following example the graph on the left uses fairly small points, the one on the right uses larger points to display the data.

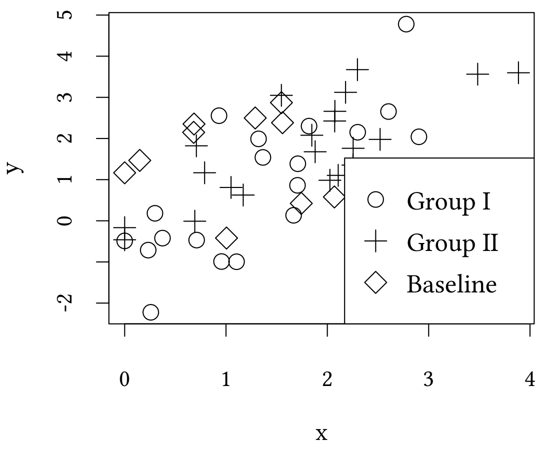
```
plot(y ~ x,cex=.25)
plot(y ~ x)
```



Shape Plotting symbols should be easy to distinguish. Compare the graph on the left with the graph on the right:

```
plot(y ~ x,pch=c(15,16,17)[z+1])
legend("bottomright",c("Group I","Group II","Baseline"),pch=c(15,16,17),bg="white")
plot(y ~ x,pch=c(1,3,5)[z+1])
legend("bottomright",c("Group I","Group II","Baseline"),pch=c(1,3,5),bg="white")
```





- When points do not tend to overlap we can use both heavy (•) and light (○) plotting symbols. Their contrasts helps us to distinguish different types of points.
- When points tend to overlap, heavy symbols are difficult to disentangle. In this situation we should use only light symbols.

Kröse's experiment

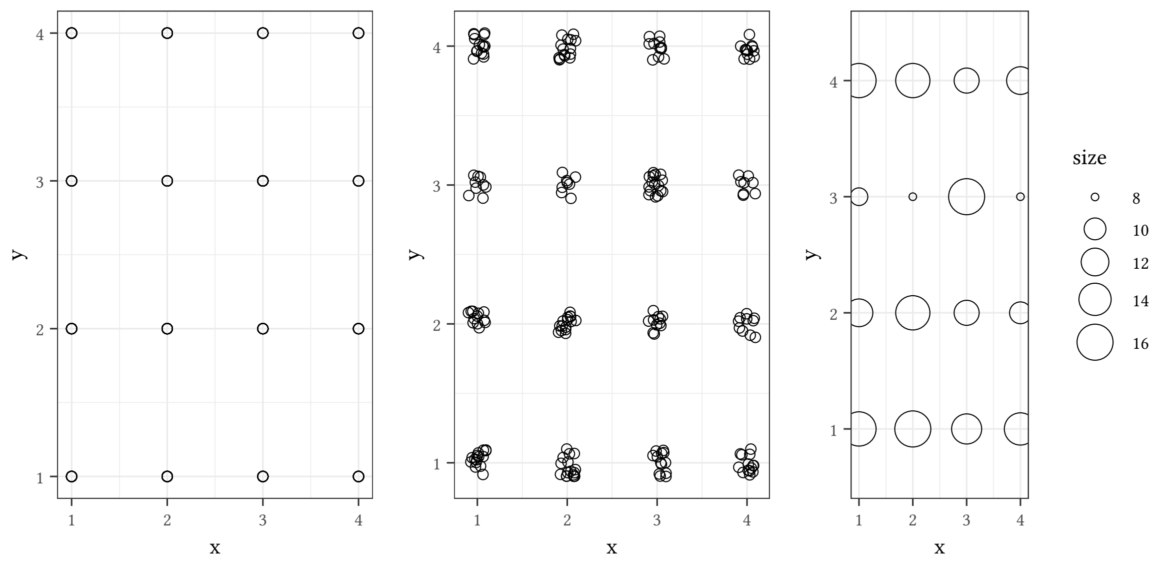
Participants see patterns of symbols for 80 ms. They have to identify presence or absence of a given symbol.

symbols	% recognised
+○	100.0
+□	88.1
L+	68.6
Δ ↓	52.3
+T	37.6
+X	30.3
TL	30.6

(B. J. A. Kröse, Local structure analyzers as determinants of preattentive pattern discrimination. Biological Cybernetics, 1987, Volume 55, Number 5, 289-298.)

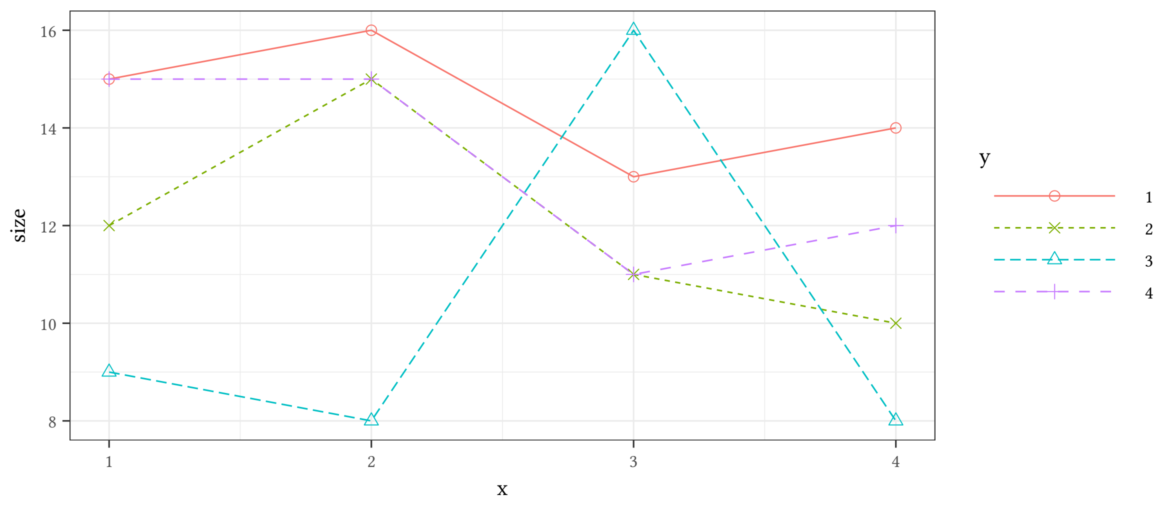
Nominal data and points Sometimes, in particular with nominal data, we want to show the same obversation several times. In the diagram on the left multiple dots are simple printed on top of each other. One can not see how frequent an observation is. In the middle we add `jitter` to each observation, small noise, which allows us to distinguish the single observations. The left graph shows frequencies as size of the symbol.

```
set.seed(1)
nomData <- data.frame(x=sample(1:4,size=200,replace=TRUE),y=sample(1:4,size=200,replace=TRUE))
p1 <- ggplot(nomData,aes(x,y)) + geom_point(shape=1)
p2 <- ggplot(nomData,aes(x,y)) + geom_jitter(width=.1,height=.1,shape=1)
nomData %>% group_by(x,y) %>% summarise(size=n()) %>% mutate(y=factor(y)) -> nomData2
p3 <- ggplot(nomData2,aes(x,y,size=size)) + geom_point(shape=1)
grid.arrange(p1,p2,p3,nrow=1)
```



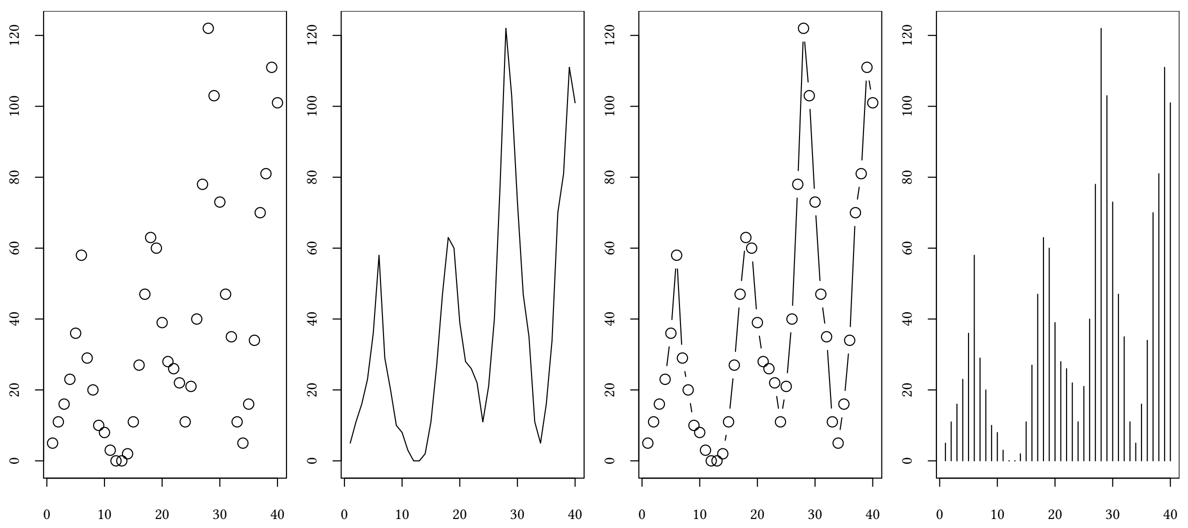
If we have a small number of categories (at least in one dimension), a dotplot might be better:

```
ggplot(nomData2,aes(x=x,y=size,color=y,lty=y,shape=y)) +
    geom_line() + geom_point() +
    theme(legend.key.width = unit(2,"cm"))
```



1.4.3 Points, lines and bars

```
plot(sunspot.year[1:40],ylab="",xlab="")
plot(sunspot.year[1:40],t="l",ylab="",xlab="")
plot(sunspot.year[1:40],t="b",ylab="",xlab="")
plot(sunspot.year[1:40],t="h",ylab="",xlab="")
```



- The development over time is easier to see with lines.
- Lines alone make it impossible to find out when the measurements were taken.

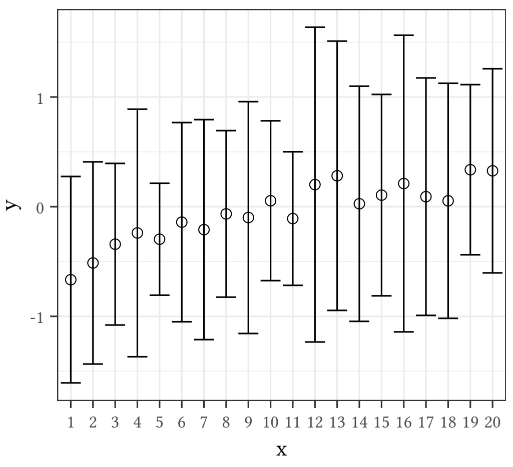
1.4.4 Error Bars

```
set.seed(6)
eBars<-data.frame(y=rnorm(200),i=rep(1:20,each=10))
eBars %>% group_by(i) %>%
    summarise(ym = median(y)+mean(y)) %>%
    mutate(x = factor(rank(ym))) %>% right_join(eBars) -> eBars2
    eBars2 %>% group_by(x) %>%
    summarise(s=sd(y),y=mean(y)) -> eBarsS
```

```
ggplot(eBarsS,aes(x=x,y=y,min=y-s,max=y+s)) + geom_point(shape=1) + geom_errorbar()
```

Error bars can refer to serveral quantities:

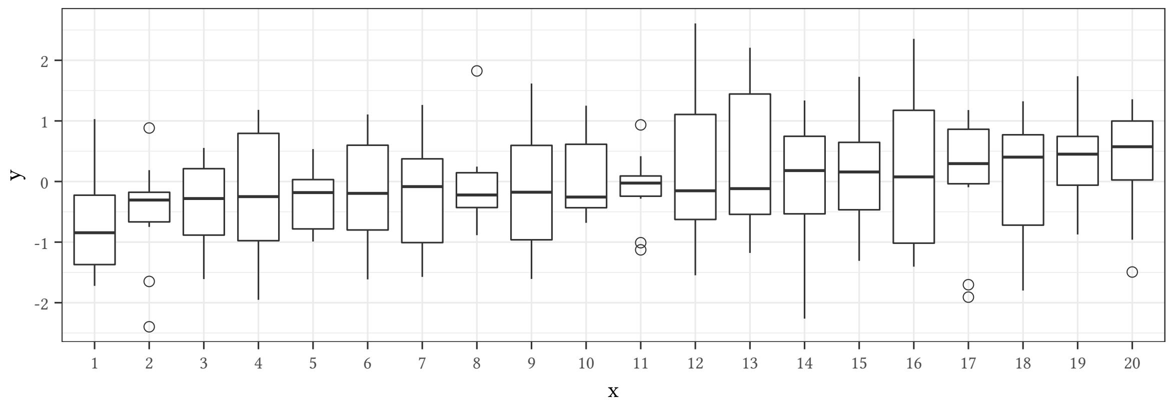
- Standard deviation of the sample
- Standard deviation of the estimated mean
- 95% confidence intervals of the estimated mean
- ⋮



We have to explain clearly in the figure which quantity (standard deviation of the sample, standard deviation of the estimated mean, confidence interval,...) is shown.

Boxplots Boxplots are often more informative than error bars.

```
ggplot(eBars2,aes(x=x,y=y)) + geom_boxplot(outlier.shape=1)
```

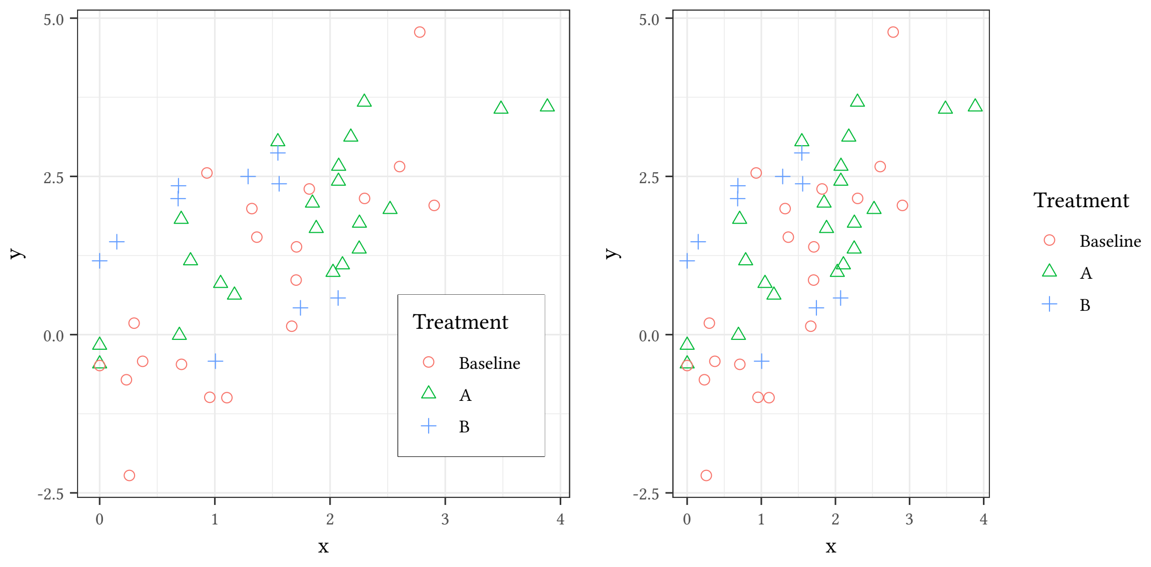


Elements of the boxplot:

- The median (thick line in the middle)
- Interquartile range (25% and 75% quantile) (the box)
- Whiskers to the most extreme data point which is no more than 1.5 times the interquartile range from the box.
- Observations outside the whiskers are shown as dots (outliers).

1.5 Legends

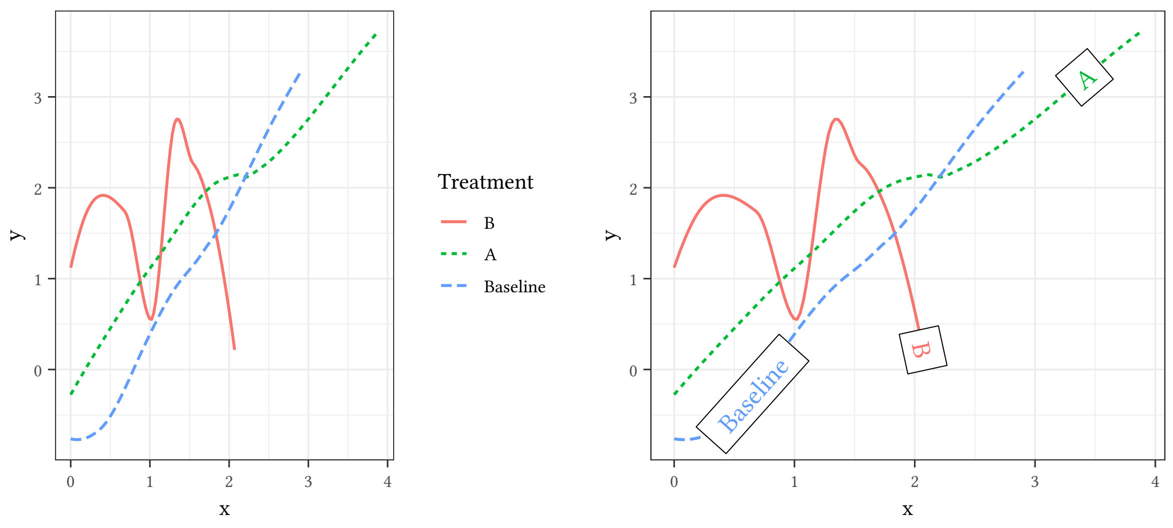
```
p1 <- ggplot(gData,aes(x=x,y=y,shape=Treatment,color=Treatment)) + geom_point() +
    scale_shape_manual(values=1:4) +
    theme(legend.position=c(.8,.25),legend.box.background = element_rect(colour = "black"))
p2 <- ggplot(gData,aes(x=x,y=y,shape=Treatment,color=Treatment)) +
    geom_point() + scale_shape_manual(values=1:4)
grid.arrange(p1,p2,nrow=1)
```



When we use more than one type of points, we need a legend. Putting the legend inside the graph saves space. However, a legend outside the graph produces less clutter (see also 1.6).

With lines, we need a legend too. It helps if labels follow the same order as lines (first graph). Often the graph is easier to understand if we label the curves directly (second graph).

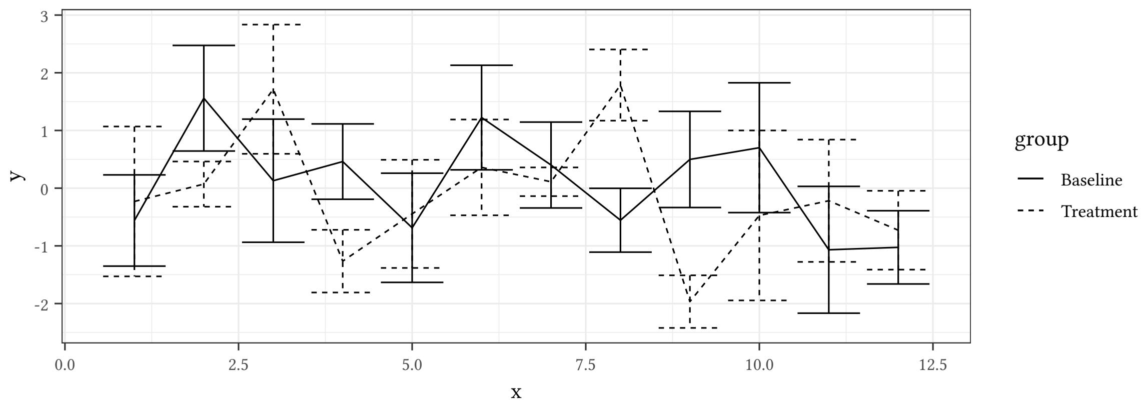
```
library(directlabels)
gData %>% mutate(Treatment=reorder(Treatment,-y,max)) %>%
    ggplot(aes(x=x,y=y,lty=Treatment,color=Treatment)) + geom_smooth(se=FALSE) -> p
grid.arrange(p,
             direct.label(p,list("far.from.others.borders","calc.boxes",
                                 "enlarge.box","draw.rects")),nrow=1)
```



1.6 Clutter

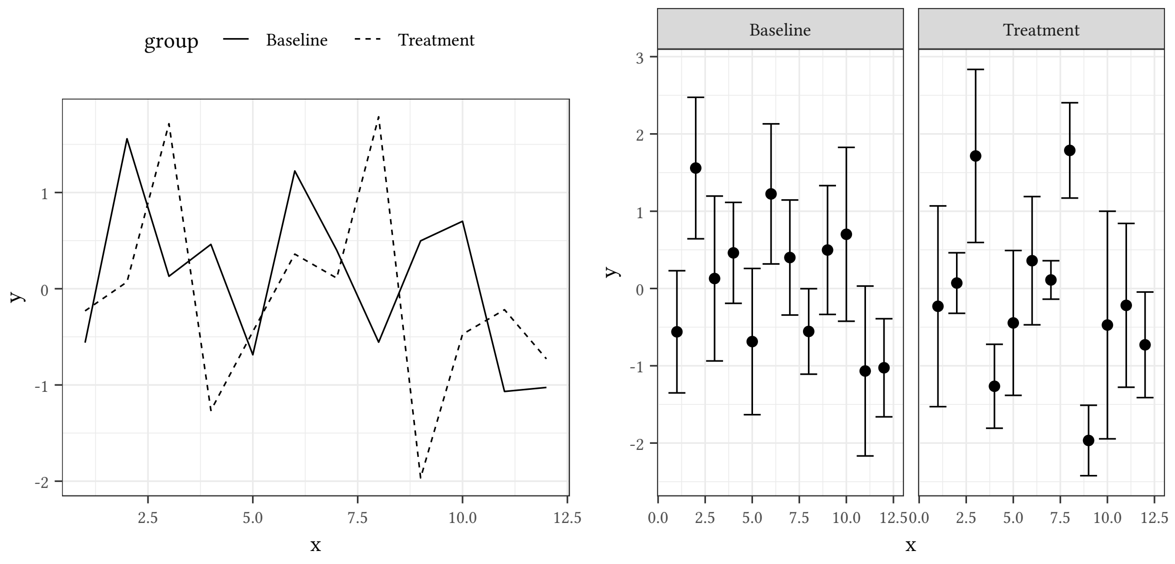
The following graph presents too many things in one graph.

```
set.seed(123)
N<-24
data.frame(y=rnorm(N),
           s=sqrt(abs(rnorm(N))),
           group=factor(rep(1:2,length.out=N),label=c("Baseline","Treatment")),
           x=rep(1:(N/2),each=2)) %>%
    mutate(ymin=y-s,
           ymax=y+s) -> dataCl

ggplot(dataCl,aes(x=x,y=y,lty=group)) + geom_line() + geom_errorbar(aes(ymin=ymin,ymax=ymax))
```

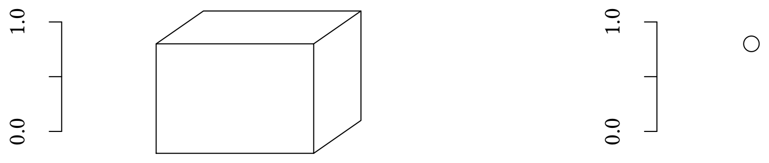


Perhaps splitting the information into several graphs can help:

```
ggplot(dataCl,aes(x=x,y=y,lty=group)) + geom_line() + theme(legend.position="top") -> p1
dataCl %>%
    ggplot(aes(x=x,y=y,ymin=ymin,ymax=ymax)) + geom_point() + geom_errorbar() +
    facet_grid(. ~ group) -> p2
grid.arrange(p1,p2,nrow=1)
```



1.7 Unnecessary 3D

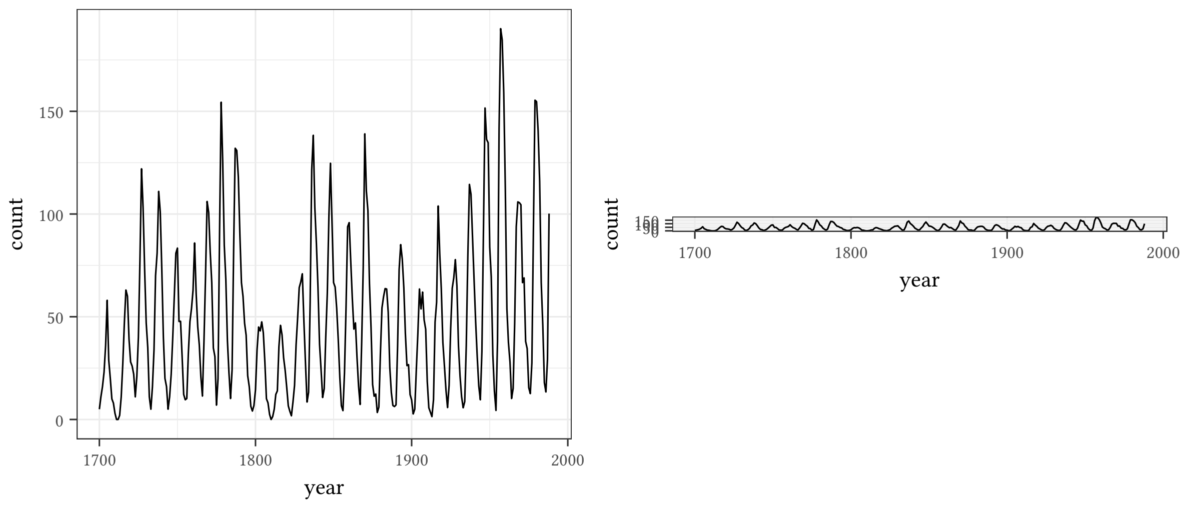


Unnecessary 3D effects distract from the content of your graph. Is the bar in the graph on the left larger or smaller than 1.0? Of course, one can work it out, but a simple dot without 3D (on the right) is much easier to understand.

1.8 Aspect ratio

Less can be more — 45°

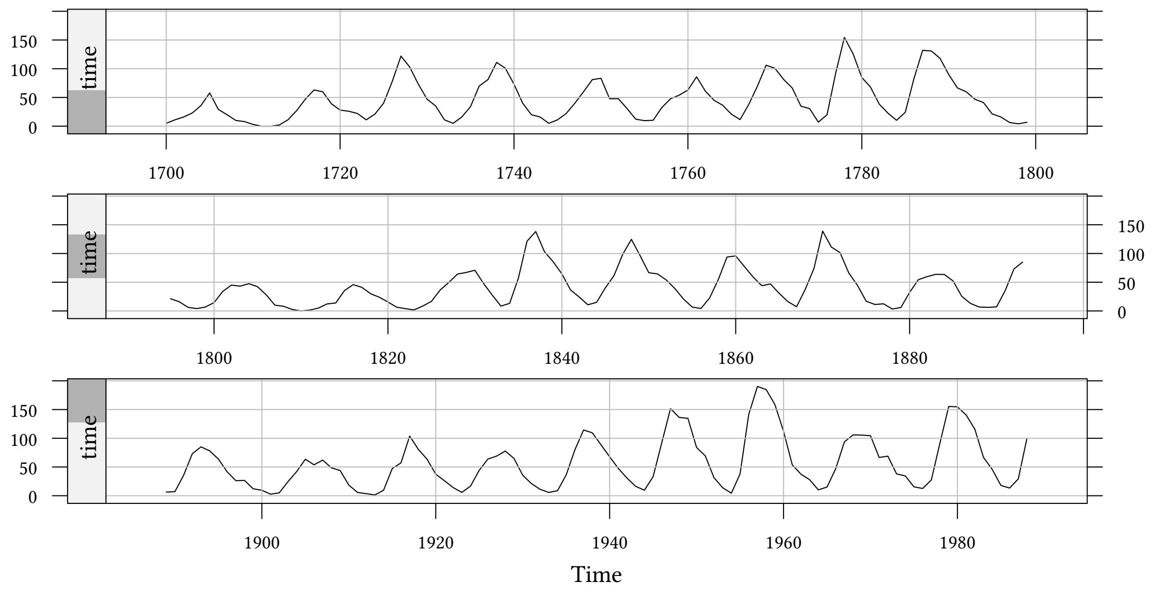
```
sunsp <- data.frame(count=as.numeric(sunspot.year),year=time(sunspot.year))
p1 <- ggplot(sunsp,aes(x=year,y=count)) + geom_line()
slope <- with(sunsp,bank_slopes(x=year,y=count))
p2 <- p1 + coord_fixed(slope)
grid.arrange(p1,p2,nrow=1)
```



The graph on the right might be more informative than the one on the left. We can see that activity increases more quickly than it decreases. This is less obvious in the left graph.

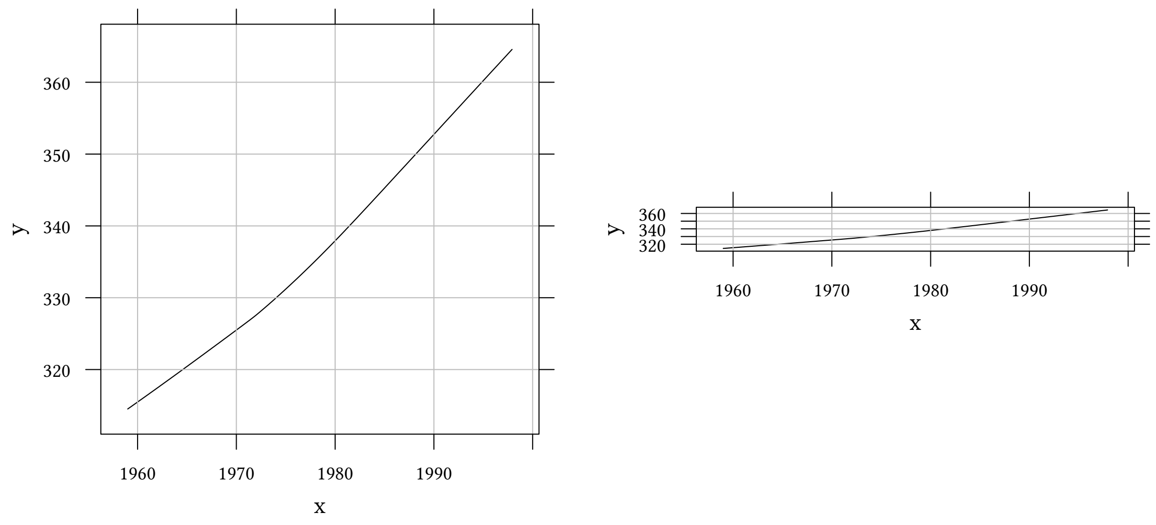
If we feel that the right graph is too flat then we can 'cut-and-stack' it as follows:

```
library(latticeExtra)
xyplot(sunspot.year,aspect="xy",strip=FALSE,strip.left=TRUE,
       cut=list(number=3,overlap=0.05))
```



In the following example the graph on the left has a slope of about 45°. This makes it easier to see the convexity of the curve.

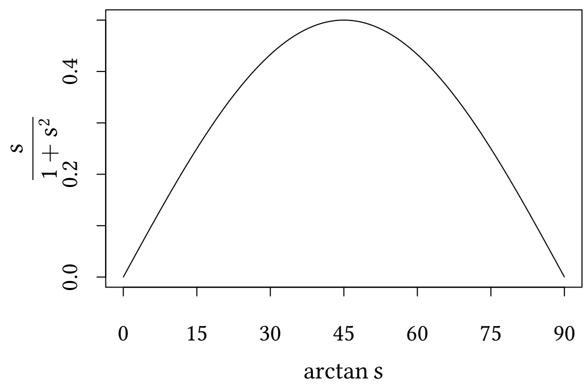
```
lco2<-data.frame(lowess(co2))
plot(xyplot(y~x,data=lco2,type="l",aspect="xy"),position=c(0,0,.5,1),more=TRUE)
plot(xyplot(y~x,data=lco2,type="l",aspect=.1),position=c(.5,0,1,1))
```



- Assume different parts of a graph have slopes s and $s \cdot (1 + \epsilon)$.
- We are interested in the differences of the slopes.

$$\Delta = \arctan s - \arctan(s \cdot (1 + \epsilon))$$

$$\frac{\partial \Delta}{\partial \epsilon} = \frac{s}{1 + (1 + \epsilon^2) \cdot s^2} \overset{\epsilon \to 0}{=} \frac{s}{1 + s^2}$$

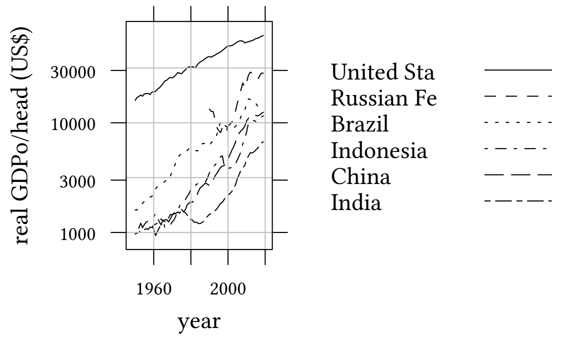


Hence, if we want to see differences in slopes, we should scale the graph such that slopes are close to 1.

Also in the following graph lines have a slope of about 45°. This makes it easier to compare the different slopes. '<

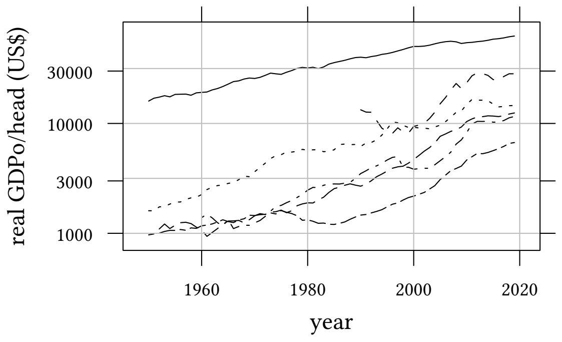
```
library(pwt10)
N<-6
data(pwt10.0)
pwt10.0 %>%
    semi_join(pwt10.0 %>% ## find N most populous countries:
              group_by(country) %>%
              summarise(popM=median(pop,na.rm=TRUE)) %>%
              arrange(-popM) %>%
              top_n(N)) %>%
    mutate(country=reorder(factor(substr(country,1,10)),-cgdpo/pop,median,na.rm=TRUE),
           gdp = cgdpo/pop) -> pwt6
```

```
xyplot(gdp ~ year,group=country,
       scales=list(y=list(log=10)),
       yscale.components=yscale.components.log10.3,
       aspect="xy",
       type="l",data=pwt6,
       auto.key=list(space="right",points=FALSE,lines=TRUE),
       ylab="real GDPo/head (US\\$)",xlab="year")
```



45° with Lattice

```
xyplot(gdp ~ year,group=country,
       scales=list(y=list(log=10)),
       yscale.components=yscale.components.log10.3,
       type="l",data=pwt6,
       ylab="real GDPo/head (US\\$)",xlab="year")
##
##
```



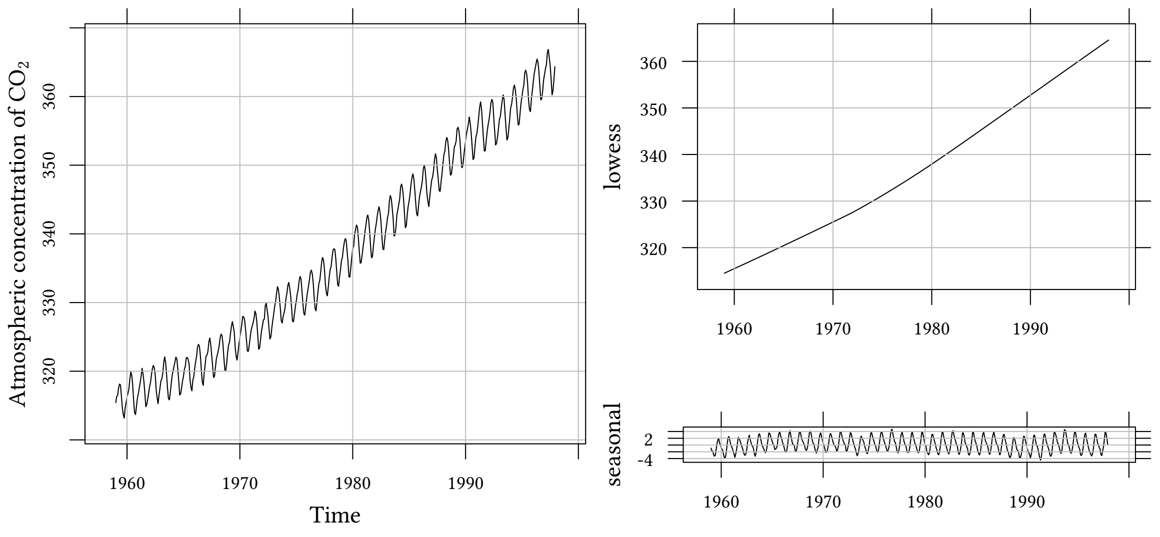
When we stretch the graph to fill the entire page, convexities and concavities are harder to see:

1.9 What to present

1.9.1 Structuring content

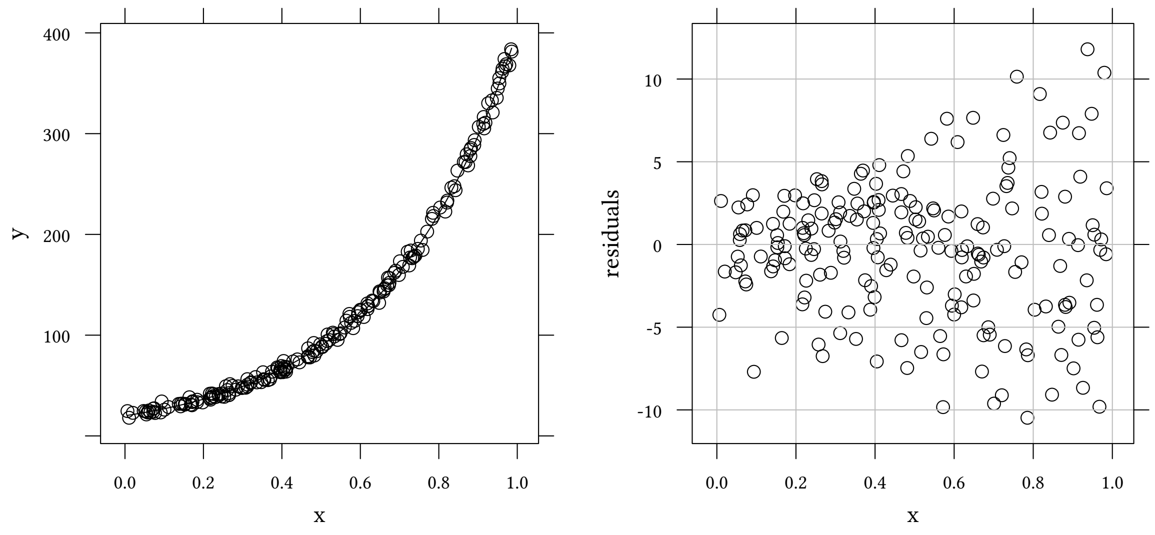
```
layout(matrix(data=c(1,1,2,3),2,2),heights=c(.7,.3))
plot(co2, ylab = "Atmospheric concentration of CO$_2$",las = 1)
lco2 <- lowess(co2)
plot(lco2,t="l",ylab="lowess",xlab="")
plot(co2-lco2$y,t='l',ylab='seasonal')
```

Mauna Loa Atmospheric CO_2 Concentration



Sometimes it helps to show residuals in a separate graph. In the following example only the graph on the right shows that noise increases for large values of x. The graph on the left does not reveal this structure.

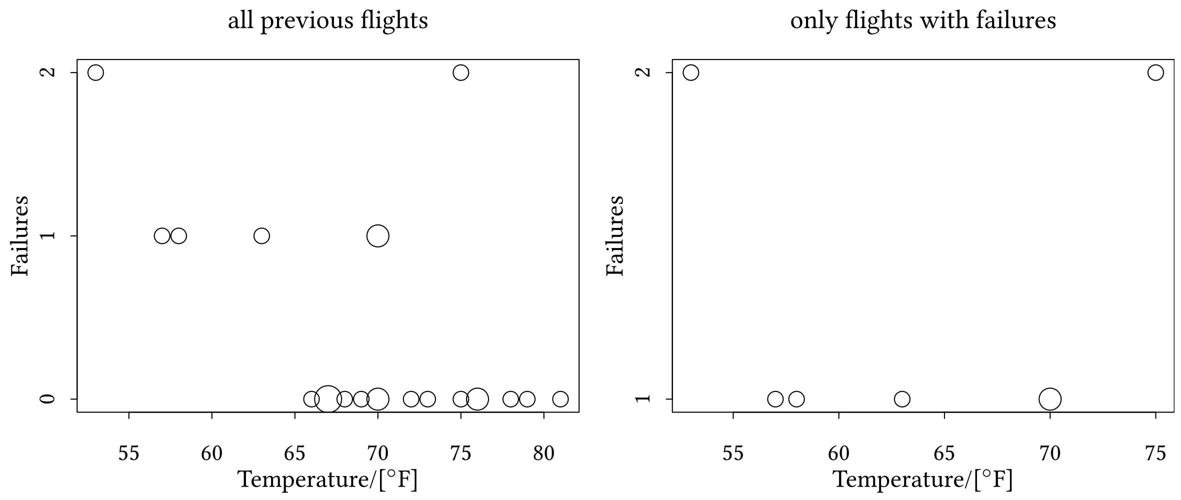
```
N <- 200
x <- sort(runif(N))
y0 <- 20 * exp(x)^3
y <- y0 +4*rnorm(N,sd=.5+x)
residuals <- y0-y
plot(lattice::xyplot(y~x,panel=function(...) {
    panel.xyplot(...);
    panel.xyplot(x=x,y=y0,type="l")}),
     position=c(0,0,.5,1),more=TRUE)
plot(xyplot(residuals ~ x),position=c(.5,0,1,1))
```



1.9.2 Don't discard parts of your data

Since temperature is measured as integers, we first have to aggregate, so that we can plot superimposed points from two different flights in a different way.

```
library(vcd)
data(SpaceShuttle)
par(mfrow=c(1,2),mar=c(4,4.5,4,0),mex=.5)
xx<-with(SpaceShuttle,aggregate(Temperature,list(Temperature=Temperature,
                                         Failures=nFailures),length))
plot(Failures ~ Temperature,data=xx,cex=sqrt(x),
     yaxt="n",xlab="Temperature/[\\degree F]",ylab="Failures",
     main="all previous flights")
axis(2,at=0:2)
plot(Failures ~ Temperature,data=xx,subset=Failures>0,cex=sqrt(x),
     yaxt="n",xlab="Temperature/[\\degree F]",ylab="Failures",
     main="only flights with failures")
axis(2,at=1:2)
```



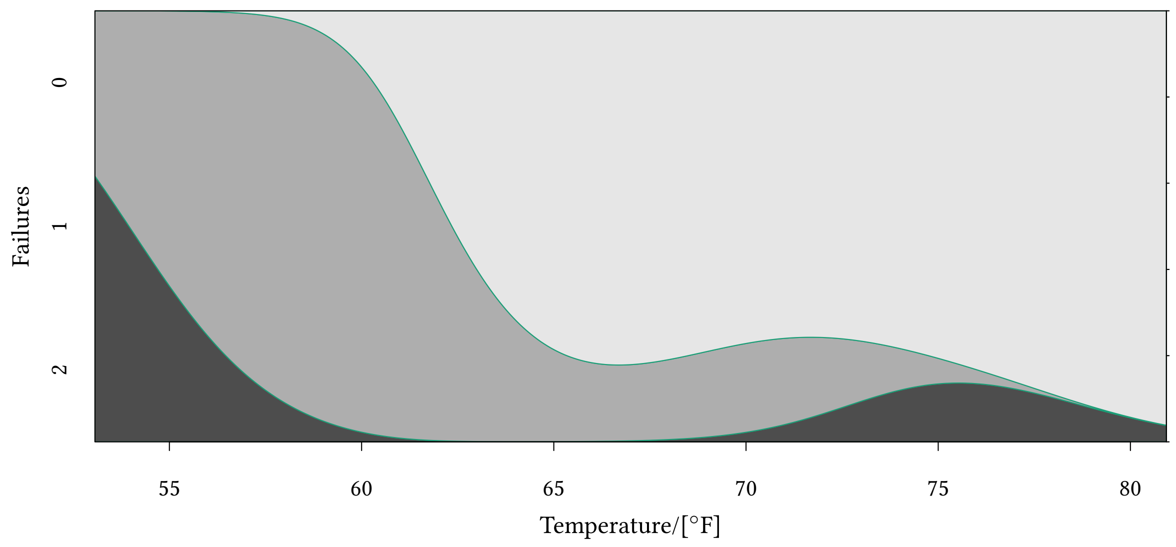
Data from S. Dalal, E. B. Fowlkes, B. Hoadly (1989).

This example illustrates the dire consequences of discarding "irrelevant" data: Previous to the crash of the space shuttle Challenger on 28 January 1986 engineers noticed that the temperature was much lower (31°F) than at other launches before (53°F to 81°F). NASA managers considered only the failures of O-rings from previous flights (diagram on the right), discarding the non-failures. They did not see any pattern in the failures and continued the countdown.[2]

Sizes of the symbols are proptional to the number of observations.

An alternative way to present this data is a conditional density plot:

```
cdplot(as.factor(nFailures) ~ Temperature,ylab="Failures",
       xlab="Temperature/[\\degree F]",data=SpaceShuttle)
```

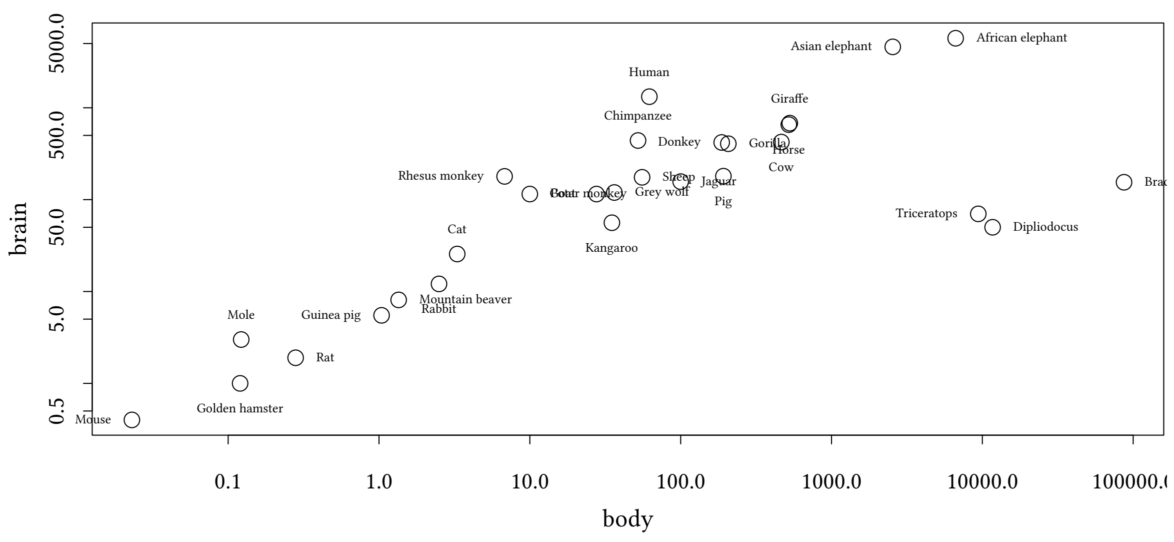


[2]Data from S. Dalal, E. B. Fowlkes, B. Hoadly (1989), Risk analysis of the space shuttle: Pre-Challenger prediction of failure, *Journal of the American Statistical Association*, *84*, 945-957.

1.9.3 Projecting data

Carl Sagan[3] argues that intelligence has something to do with the weight of the brain and the weight of the body. We are supposed to see this from a graph which is similar to the following:

```
library(MASS)
data(Animals)
plot(brain ~ body,data=Animals,log="xy")
with(Animals,thigmophobe.labels(body,brain,rownames(Animals),cex=.5))
```



This is too complicated. Stephen Jay Gould[4]:

- brain size should be proportional to the surface of the body
- surface grows quadratically with height volume (and weight) grows cubically

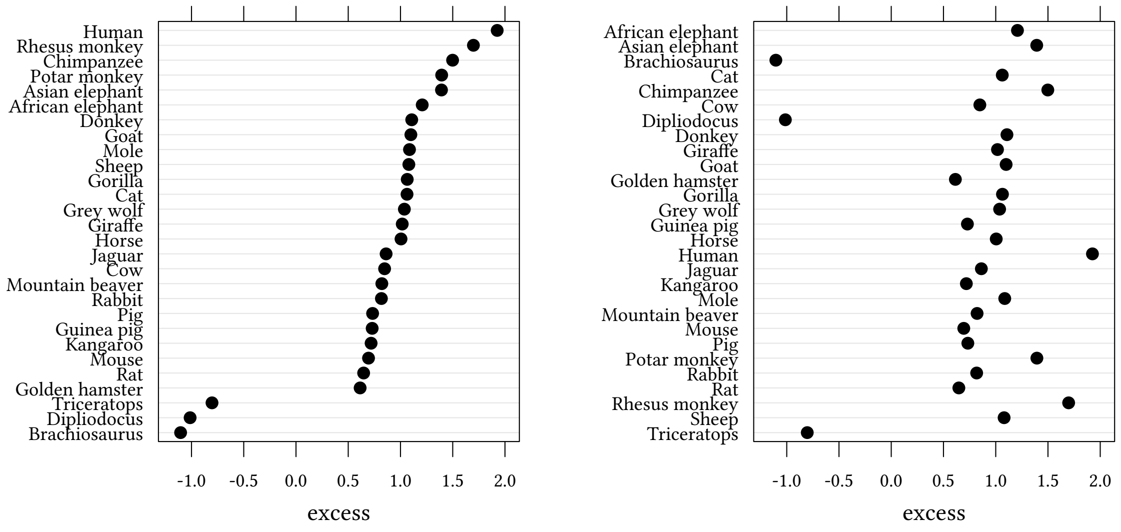
$\rightarrow$ weight of the brain $\sim$ weight of the body$^{2/3}$

$$\text{excess brain} = \log(\text{brain mass}) - \frac{2}{3}\log(\text{body mass})$$

To make it easier to interpret this difference of logs we use logarithms with base 10.
The left graph is ordered by the quantity of “excess brain”, the right one is ordered alphabetically. Often dotplots are easier to understand when they are sorted by the quantity.

[3]The Dragons of Eden: Speculations on the Evolution of Human Intelligence. Random House, New York, 1977.
[4]Ever Since Darwin: Reflections in Natural History. Norton, New York, 1977.

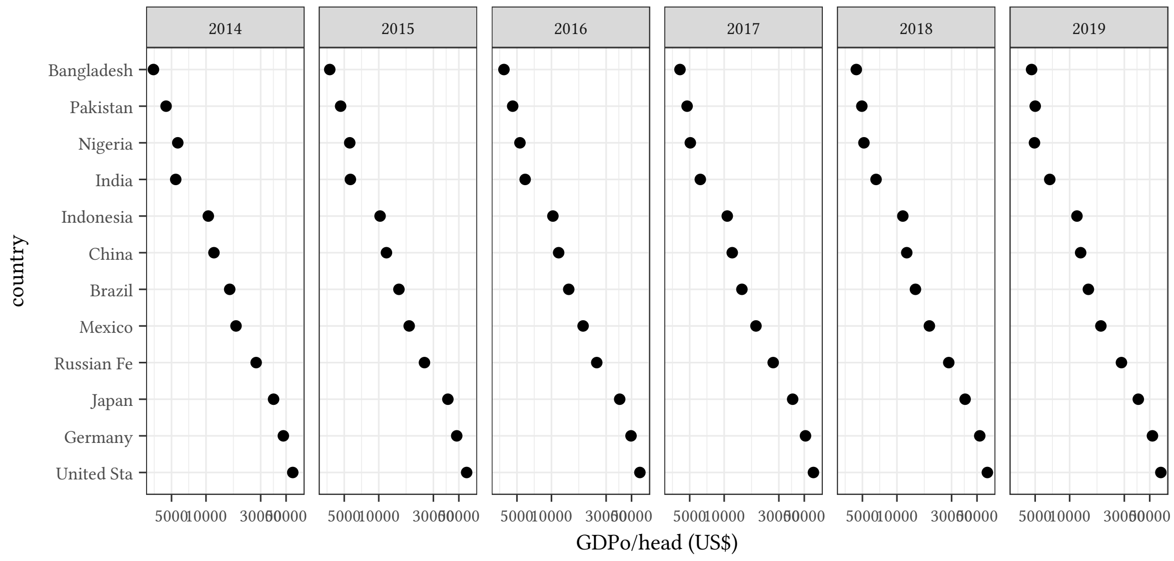
```
excess <- with(Animals,log10(brain)-2/3*log10(body))
xx<-data.frame(list(Animals=reorder(rownames(Animals),excess,median),excess=excess))
library(lattice)
plot(dotplot(Animals ~ excess,data=xx),more=TRUE,position=c(0,0,.5,1))
xx<-data.frame(list(Animals=reorder(rownames(Animals),
                    -as.numeric(factor(rownames(Animals))),median),excess=excess))
plot(dotplot(Animals ~ excess,data=xx),position=c(.5,0,1,1))
```

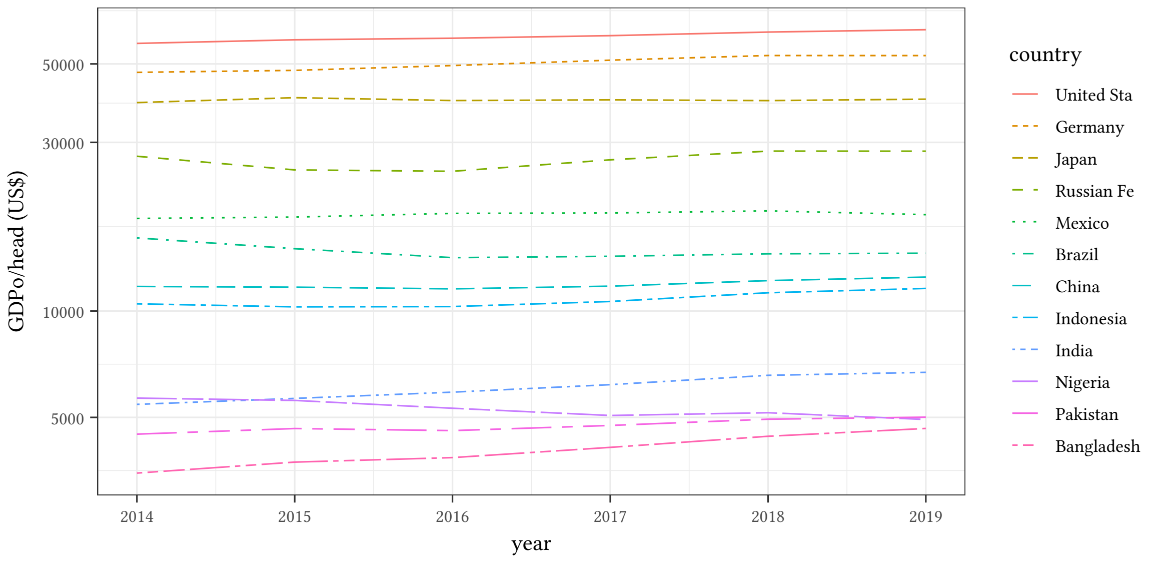


Also for a multiway dotplot ordering by quantity helps. In the following example we use medians of the different categories.

```
data(pwt10.0)
N <- 12
pwt10.0 %>%
    semi_join(pwt10.0 %>% ## find N most populous countries:
              group_by(country) %>%
              summarise(popM=median(pop,na.rm=TRUE)) %>%
              arrange(-popM) %>%
              top_n(N)) %>%
    filter(year>max(year)-6) %>%
    mutate(gdp = cgdpo/pop,
           country = reorder(factor(substr(country,1,10)),-gdp,median,na.rm=TRUE)) -> pwt12
```

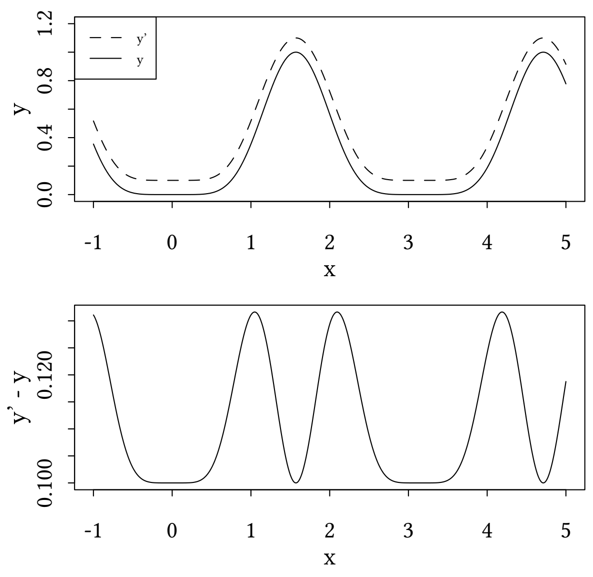
```
ggplot(pwt12,aes(y=country,x=gdp)) + geom_point() + facet_wrap(vars(year),nrow=1) +
    scale_x_log10()  + labs(x="GDPo/head (US\\$)")
ggplot(pwt12,aes(y=gdp,x=year,color=country,lty=country)) + geom_line() +
    scale_y_log10()  + labs(y="GDPo/head (US\\$)")
```





1.9.4 Differences

```
par(mfrow=c(2,1),mex=.5)
x <- seq(-1,5,.01)
y <- sin(x)^6
dy <- abs(6*cos(x)^2*sin(x)^6)
plot(y ~ x,t="l",ylim=c(0,1.2))
lines(x,y+.1*dy+.1,lty=2)
legend("topleft",c("y'","y"),lty=c(2,1),cex=.5)
plot(x,+.05*dy+.1,t="l",ylab="y' - y",xlab="x")
```



In the top graph it is difficult to assess the difference between the two curves.

If it is the difference that is interesting, then then graph should show the difference (bottom graph).

2 Graphs with `ggplot2`

R provides a number of ways to create graphs. The most basic is perhaps the built in `plot`. More powerful ones are `lattice` and `ggplot2`. Here we use `ggplot2` as a starting point. In this chapter we want to explain how some standard graphs can be created with `ggplot2`.

2.1 Elements of ggplot

The iris data For our examples we need some data. One standard data set is the iris data.

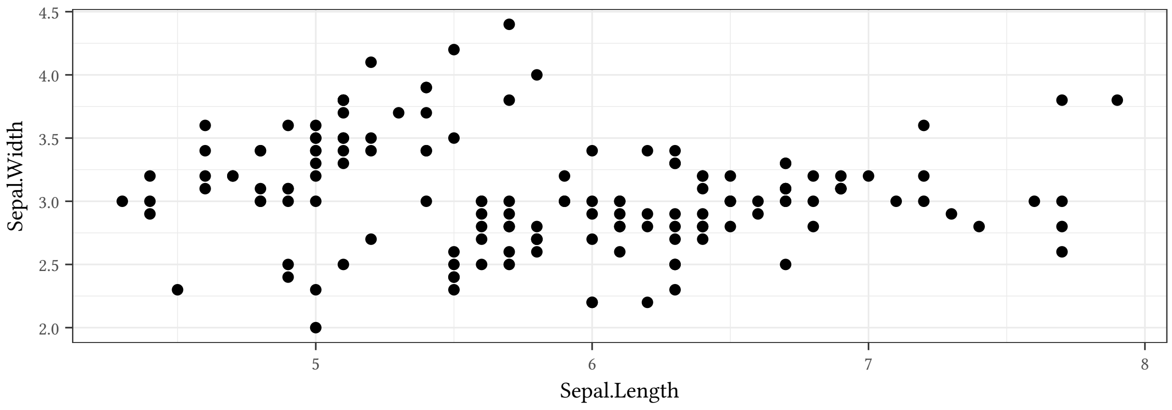
```
iris
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
7          4.6         3.4          1.4         0.3  setosa
8          5.0         3.4          1.5         0.2  setosa
 [ reached 'max' / getOption("max.print") -- omitted 142 rows ]
```

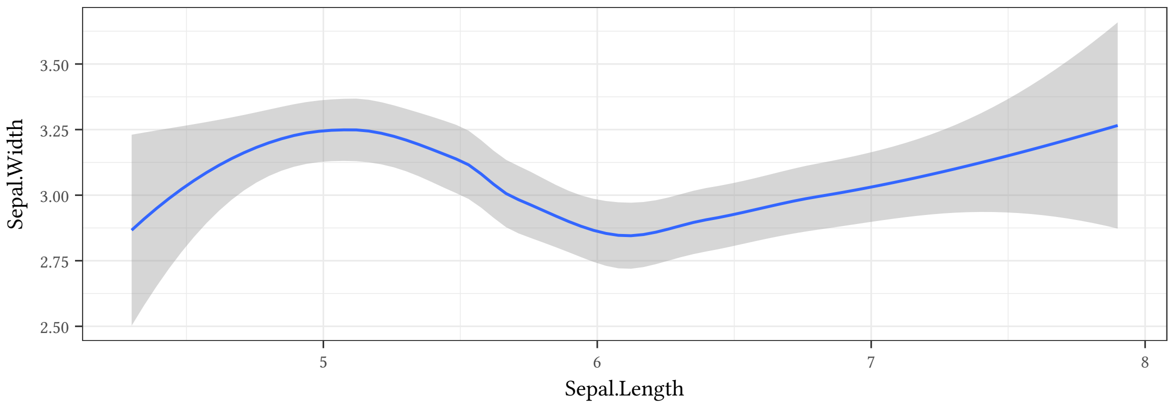
Anderson, Edgar (1935). The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, 59, 2-5.

aes and geom:

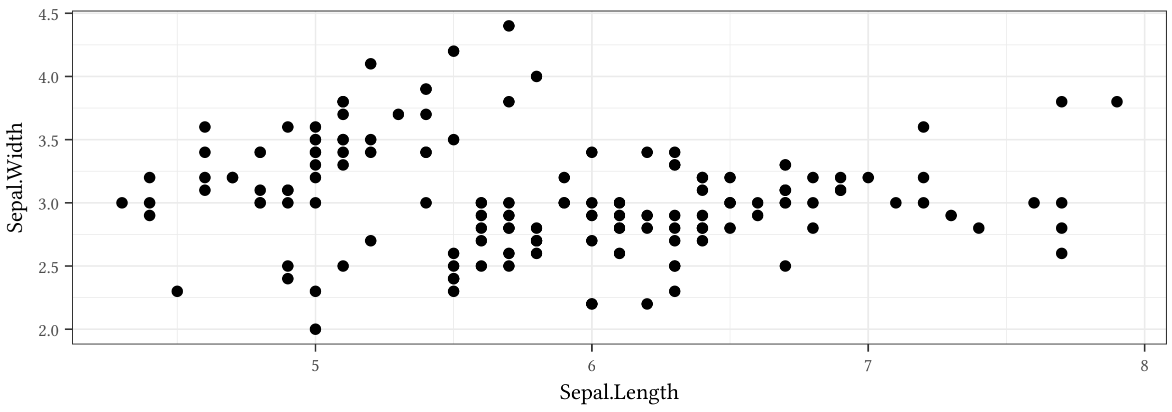
```
library(ggplot2)
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_point()
```



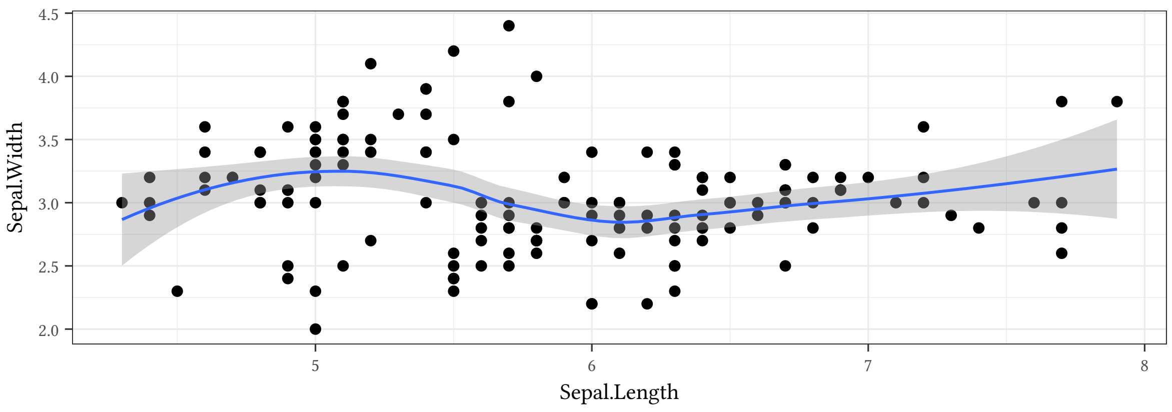
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_smooth()
```



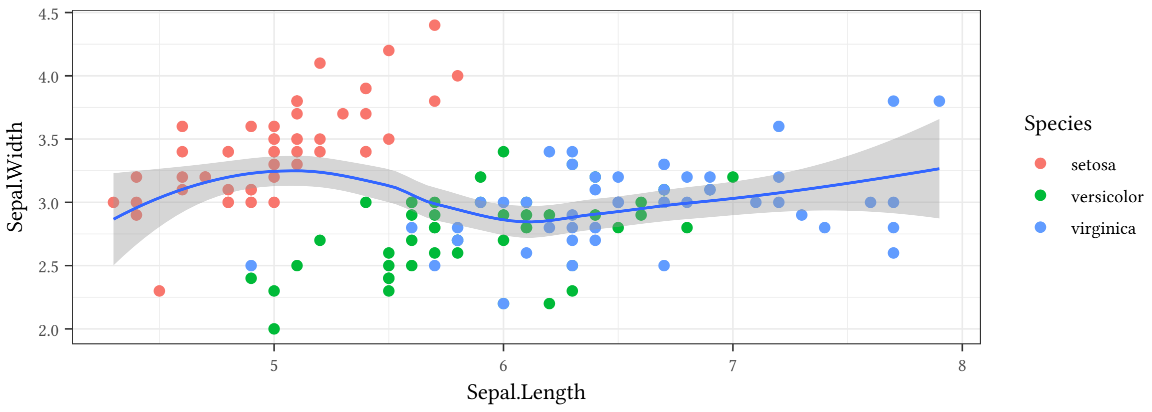
```
ggplot(iris) +
    geom_point(aes(x=Sepal.Length,y=Sepal.Width))
```



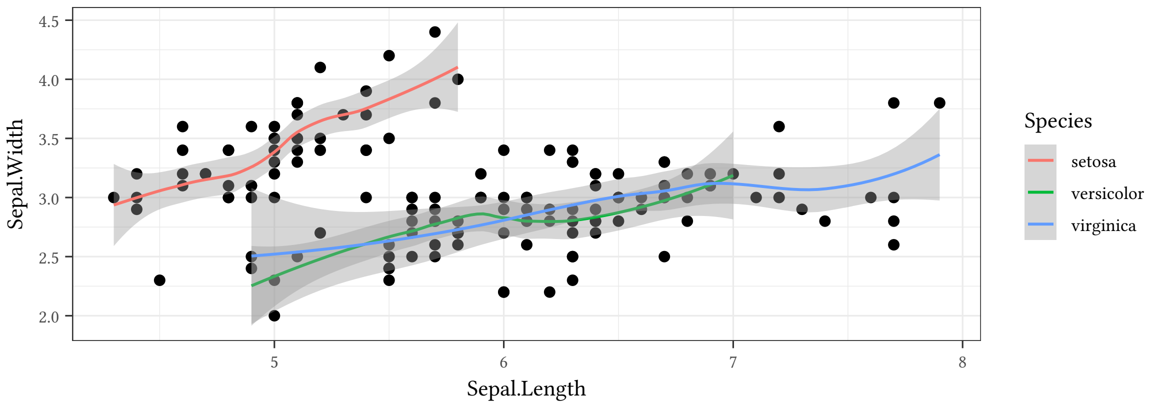
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_point() + geom_smooth()
```



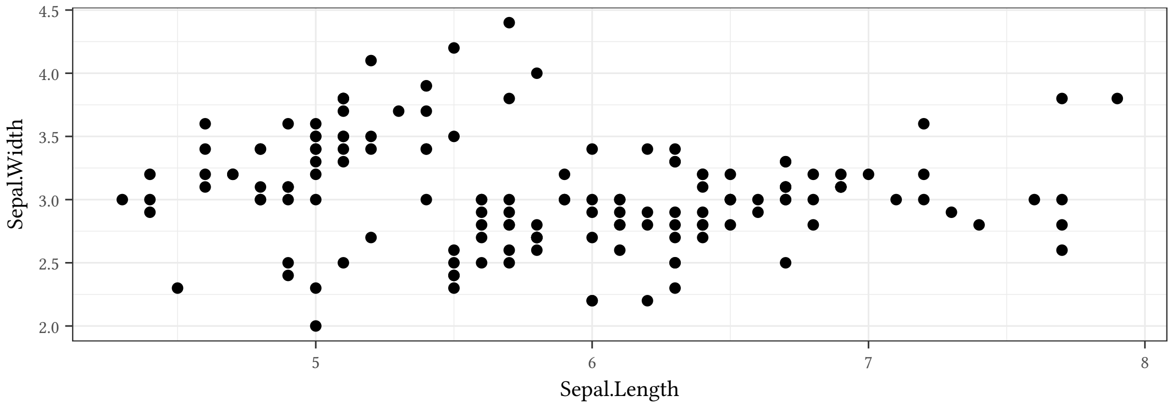
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_point(aes(color=Species)) + geom_smooth()
```



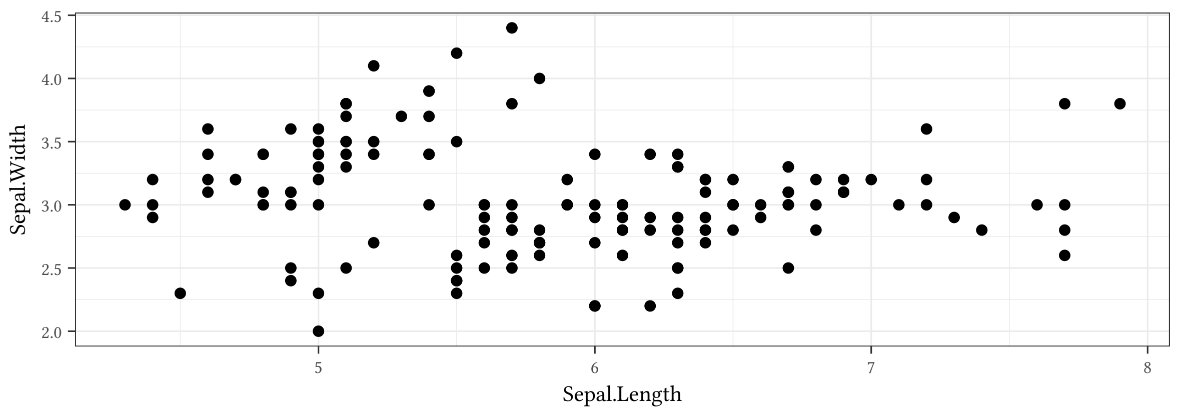
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_point() + geom_smooth(aes(color=Species))
```



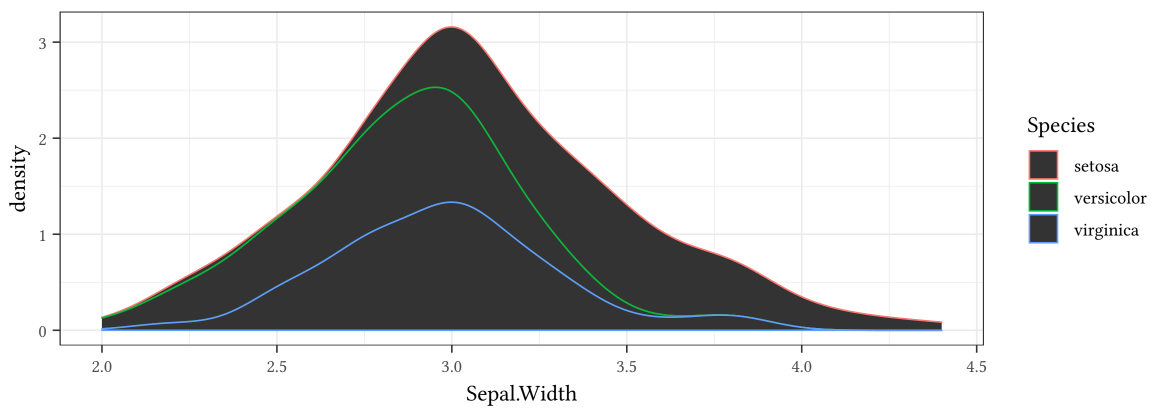
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    stat_identity(geom="point")
```



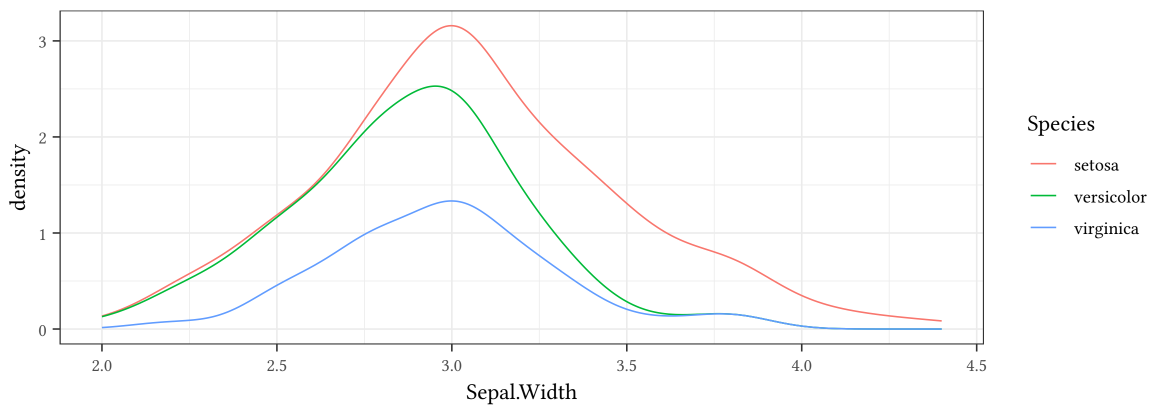
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_point(stat="identity")
```



```
ggplot(iris,aes(x=Sepal.Width,color=Species)) +
    stat_density()
```



```
ggplot(iris,aes(x=Sepal.Width,color=Species)) +
    stat_density(geom="line")
```



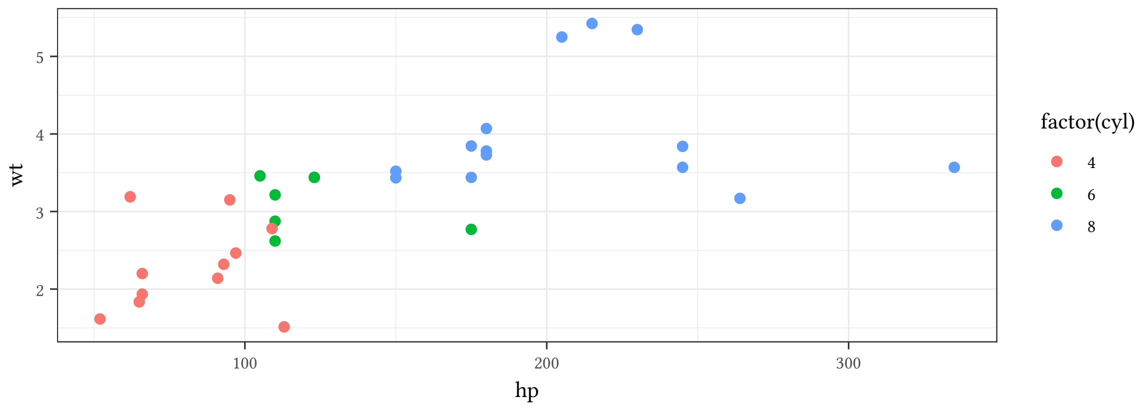
2.2 Labels and legends

```
mtcars
```

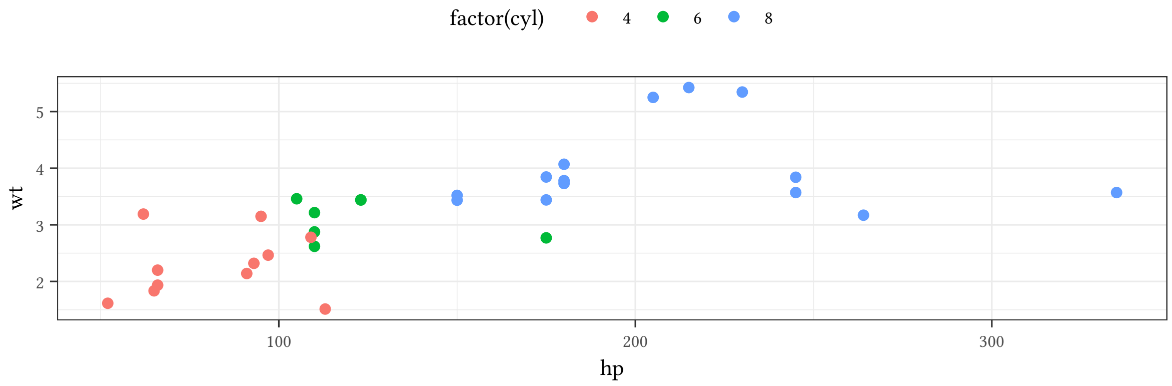
```
               mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
 [ reached 'max' / getOption("max.print") -- omitted 29 rows ]
```

Data from *Motor Trend*. 1974.

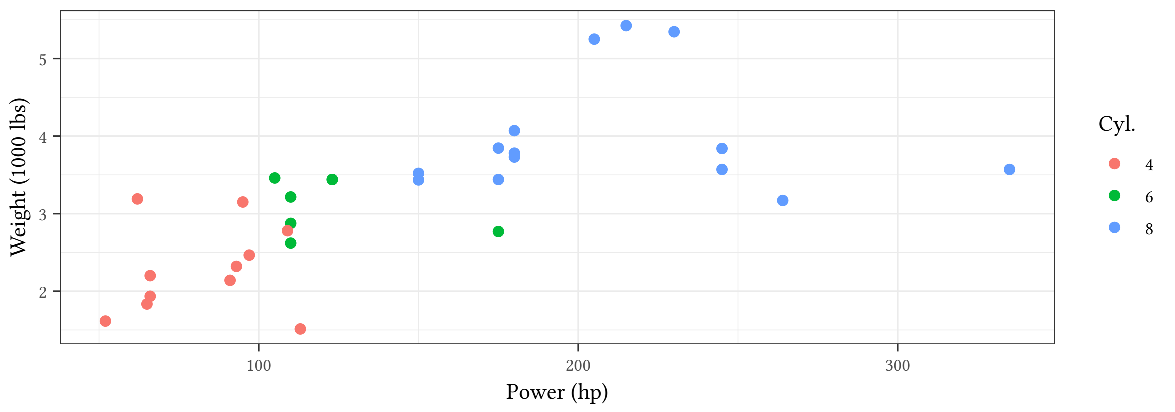
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl))) +
    geom_point()
```



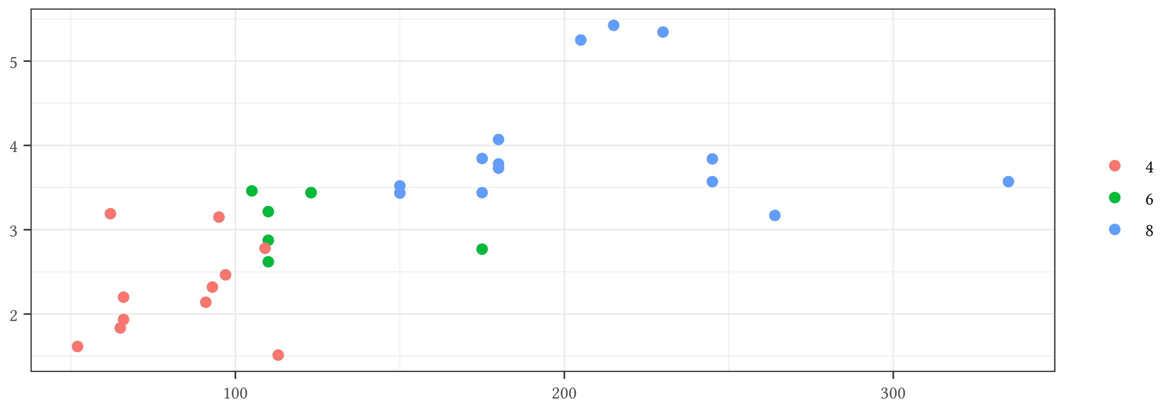
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl))) +
    geom_point() + theme(legend.position="top")
```



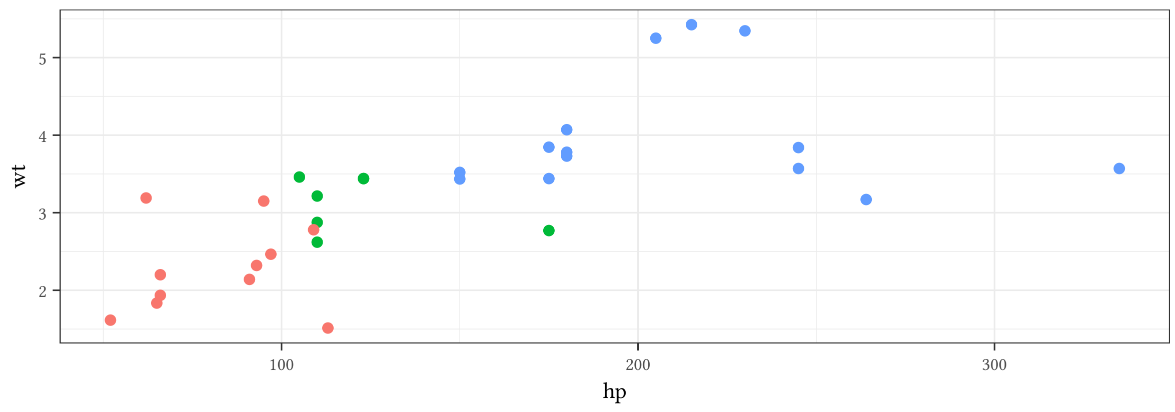
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl))) +
    geom_point() +
    labs(color="Cyl.",x="Power (hp)",
         y="Weight (1000 lbs)")
```



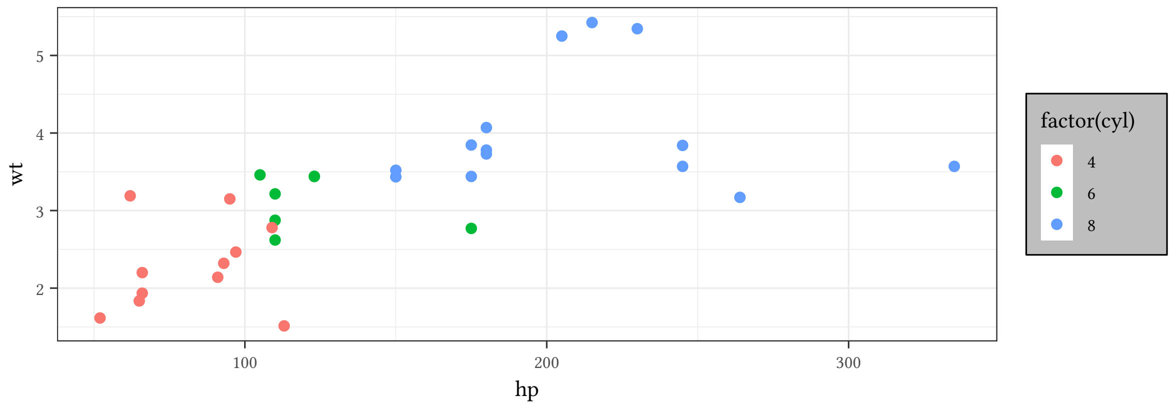
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl))) +
    geom_point() +
    labs(color=NULL,x=NULL,y=NULL)
```



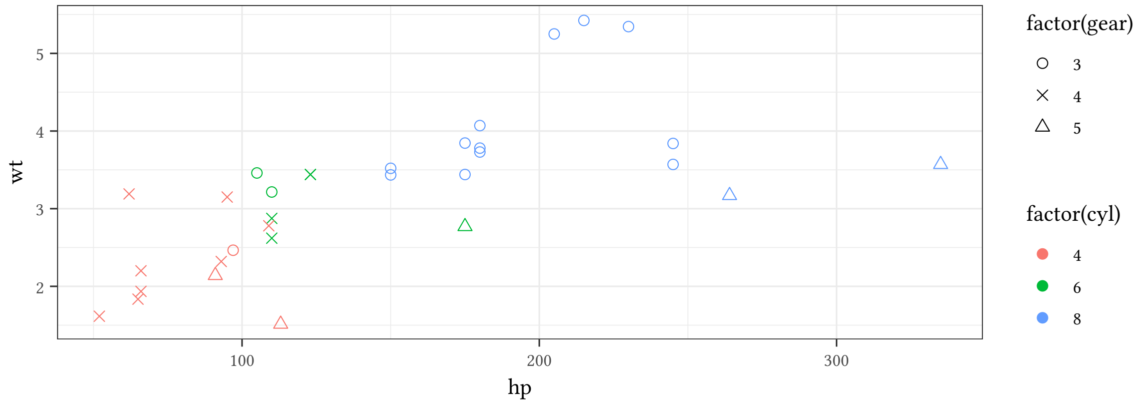
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl))) +
    geom_point() +
    theme(legend.position="none")
```



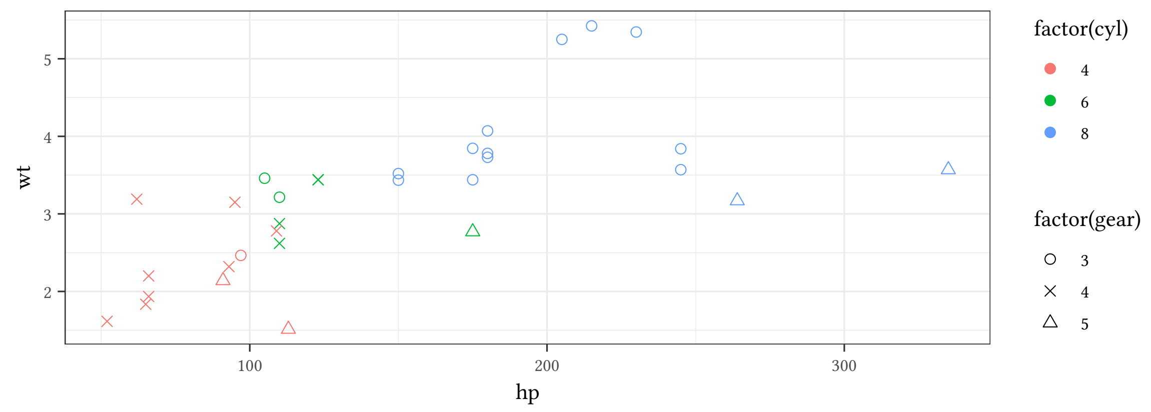
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl))) +
    geom_point() +
    theme(legend.background=element_rect(
              fill="gray",color="black"))
```



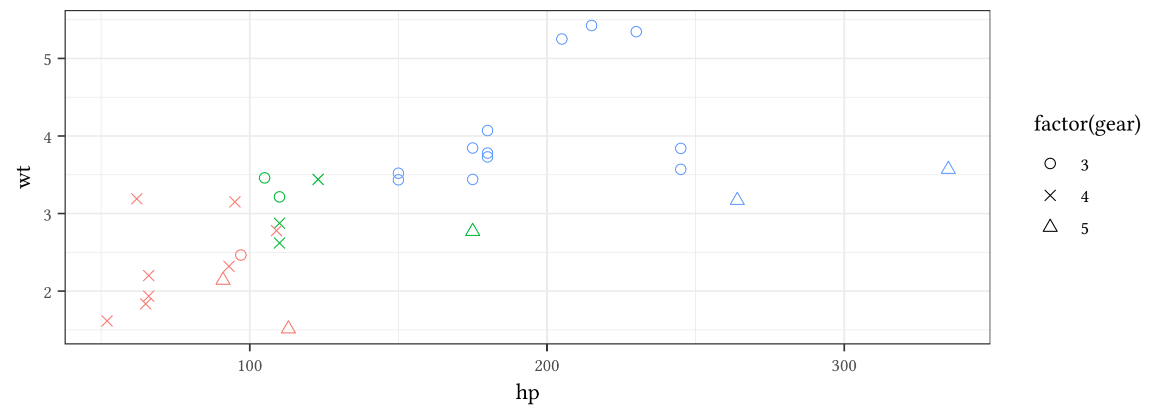
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl),
                  shape=factor(gear))) +
    geom_point()
```



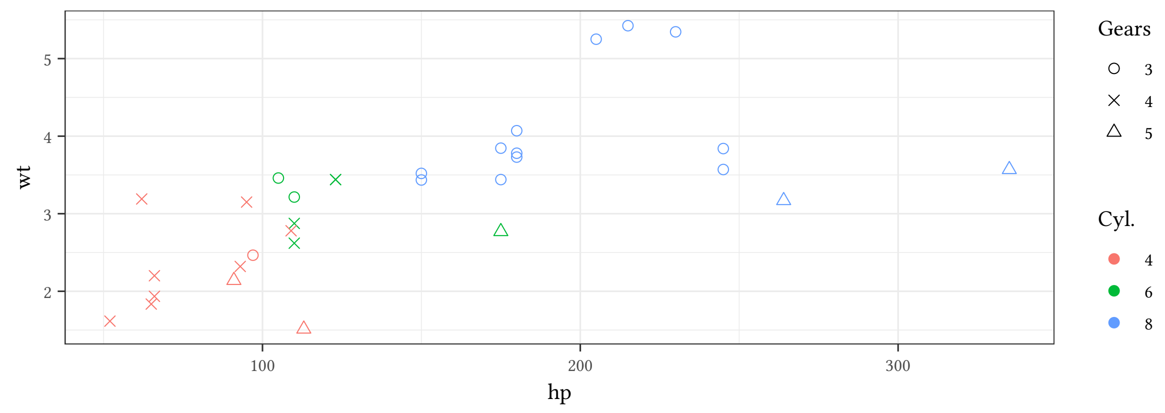
```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl),
                  shape=factor(gear))) +
    geom_point() +
    guides(color=guide_legend(order=1))
```



```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl),
                  shape=factor(gear))) +
    geom_point() + guides(color="none")
```



```
ggplot(mtcars,aes(x=hp,y=wt,color=factor(cyl),
                  shape=factor(gear))) +
    geom_point() +
    labs(shape="Gears",color="Cyl.")
```



2.3 Scatterplots

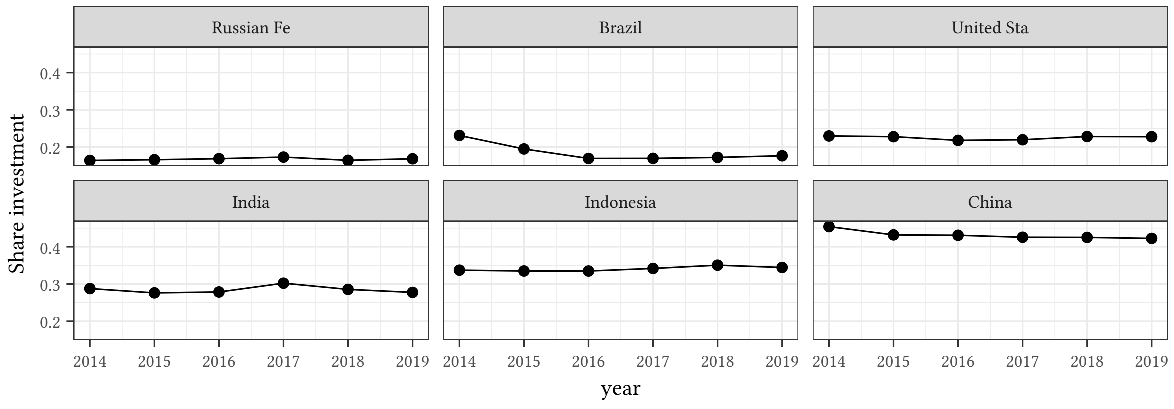
```
library(pwt10)
data(pwt10.0)
```

```
library(dplyr)
pwtYC <- function(years,countries) {
    pwt10.0 %>%
        semi_join(pwt10.0 %>% ## find N most populous countries:
                    group_by(country) %>%
                    summarise(popM=median(pop,na.rm=TRUE)) %>%
                    arrange(-popM) %>%
                    top_n(countries)) %>%    ## only the ... largest countries
        filter(year>max(year)-years) %>% ## only the last ... years
        mutate(gdp = cgdpo/pop,
               country = substr(country,1,10)) %>%
        select(c("country","gdp","year","csh_c","csh_i","csh_g"))
}
```

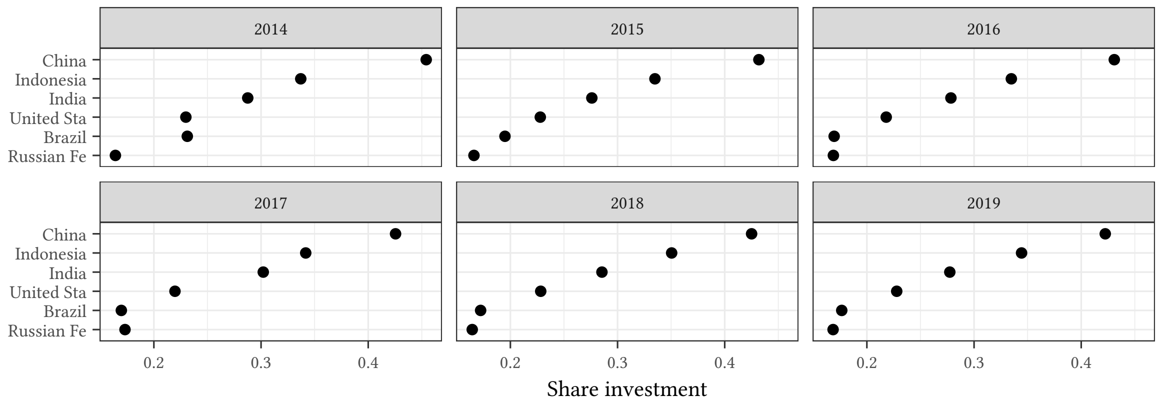
Feenstra RC, Inklaar R, Timmer MP (2015). The Next Generation of the Penn World Table, *American Economic Review*, 105(10). pp. 3150-82.

```
pwtYC(years=6, countries=6)


         country      gdp year     csh_c     csh_i     csh_g
BRA-2014  Brazil 16099.69 2014 0.6338230 0.2311709 0.1671210
BRA-2015  Brazil 15005.83 2015 0.6415980 0.1949314 0.1758238
BRA-2016  Brazil 14154.57 2016 0.6427402 0.1694520 0.1856983
BRA-2017  Brazil 14279.43 2017 0.6361969 0.1696125 0.1872893
BRA-2018  Brazil 14514.13 2018 0.6432136 0.1722161 0.1847648
BRA-2019  Brazil 14570.64 2019 0.6462484 0.1765482 0.1824011
 [ reached 'max' / getOption("max.print") -- omitted 30 rows ]
```

```
ggplot(data=pwtYC(years=6, countries=6), aes(x=year,y=csh_i)) +
    geom_line() + geom_point() +
    facet_wrap( ~ reorder(country,csh_i)) +
    labs(y="Share investment")
```

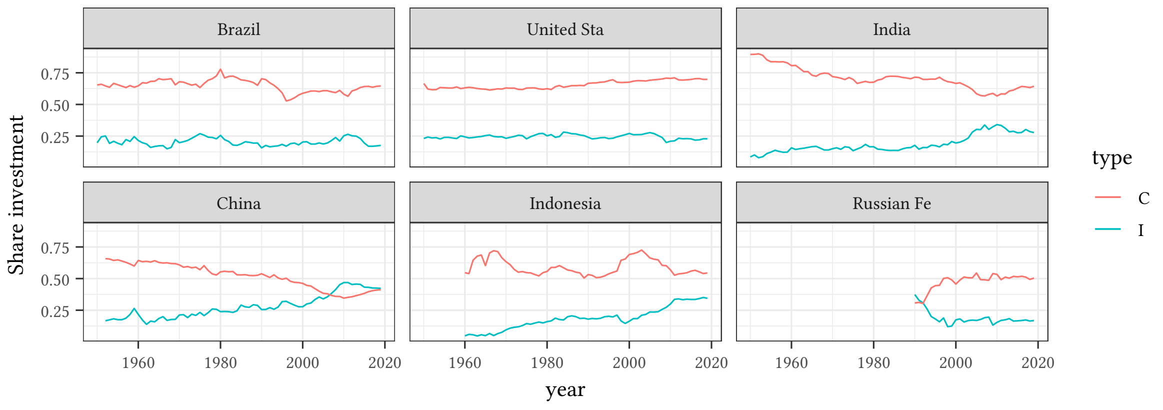


```
ggplot(data=pwtYC(years=6, countries=6), aes(y=reorder(country,csh_i),x=csh_i)) +
    geom_point() +
    facet_wrap( vars(year) ) +
    labs(x="Share investment",y=NULL)
```



We can simply draw (within ggplot) two lines with two geoms: ...as several geoms:

```
pwtYC(99,6) %>% ggplot(aes(x=year,y=csh_i)) +
    geom_line(aes(color="I")) +
    geom_line(aes(y=csh_c,color="C")) +
    facet_wrap(~reorder(country,csh_c)) +
    labs(y="Share investment",color="type")
```



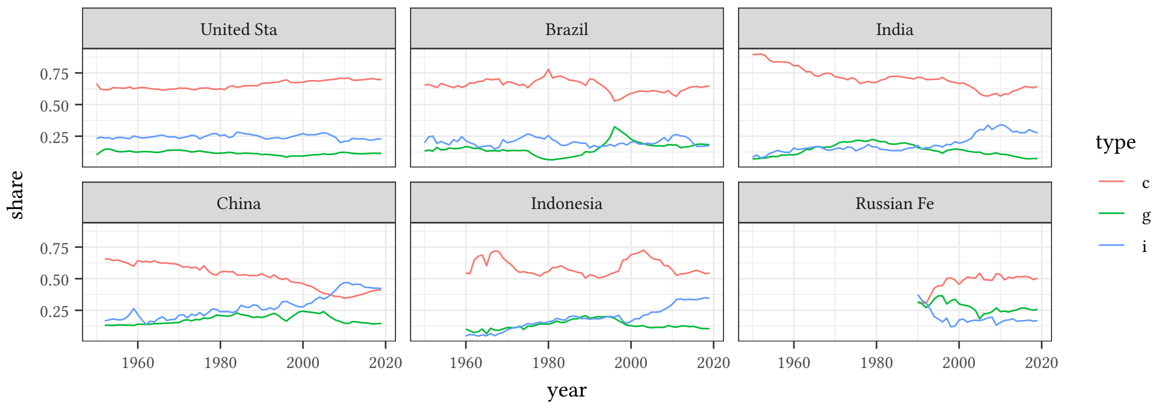
Alternatively we reshape the data before ggplot.

```
pwtYC(99,6) %>% head(n=3)

         country      gdp year      csh_c     csh_i     csh_g
BRA-1950  Brazil 1606.117 1950 0.6536519 0.1972710 0.1327842
BRA-1951  Brazil 1602.992 1951 0.6593236 0.2439667 0.1399094
BRA-1952  Brazil 1739.890 1952 0.6463201 0.2511176 0.1322049
```

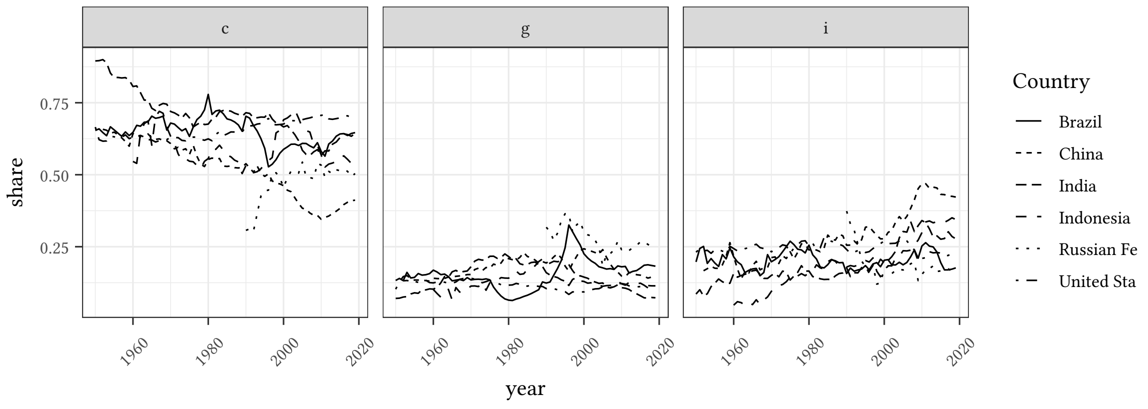
```
pwtYC(99,6) %>% tidyr::pivot_longer(cols=starts_with("csh"),
                                    names_to="type",
                                    names_prefix="csh_",
                                    values_to="share") -> pwtLong
```

```
pwtLong %>% slice_head(n=3)

# A tibble: 3 x 5
  country   gdp  year type  share
  <chr>   <dbl> <int> <chr> <dbl>
1 Brazil  1606.  1950 c     0.654
2 Brazil  1606.  1950 i     0.197
3 Brazil  1606.  1950 g     0.133
```

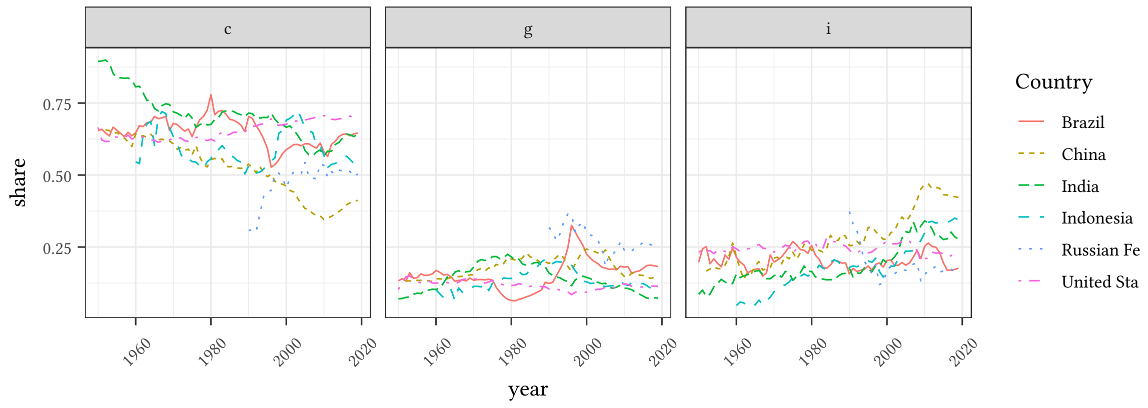
```
pwtLong %>% ggplot(aes(x=year,y=share,color=type)) +
    facet_wrap(~reorder(country,share,max)) +
    geom_line()
```



```
pwtLong %>% ggplot(aes(x=year,y=share,lty=country)) +
    geom_line() + facet_wrap(~type) + labs(lty="Country") +
    theme(axis.text.x = element_text(angle=45,vjust=.5))
```



```
pwtLong %>% ggplot(aes(x=year,y=share,lty=country,color=country)) +
    geom_line() + facet_wrap(~type) + labs(color="Country",lty="Country") +
    theme(axis.text.x = element_text(angle=45,vjust=.5))
```

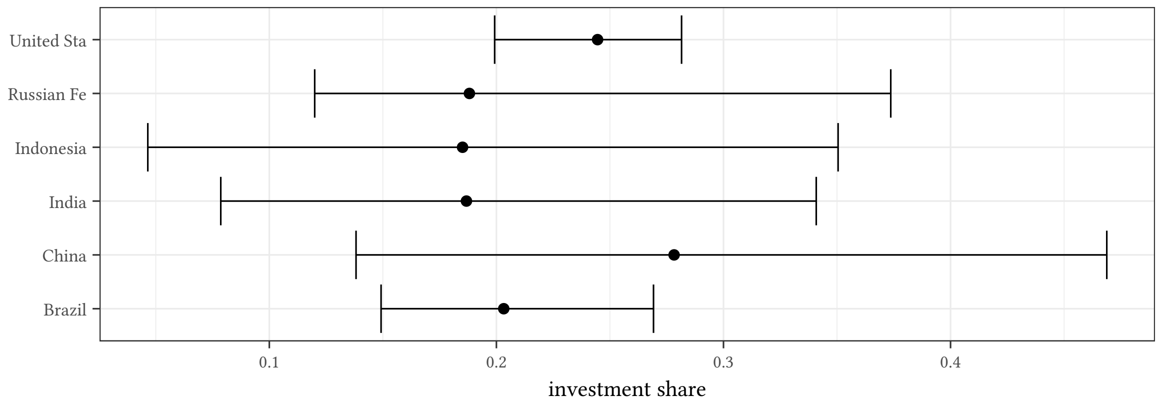


3 More graphs with ggplot2

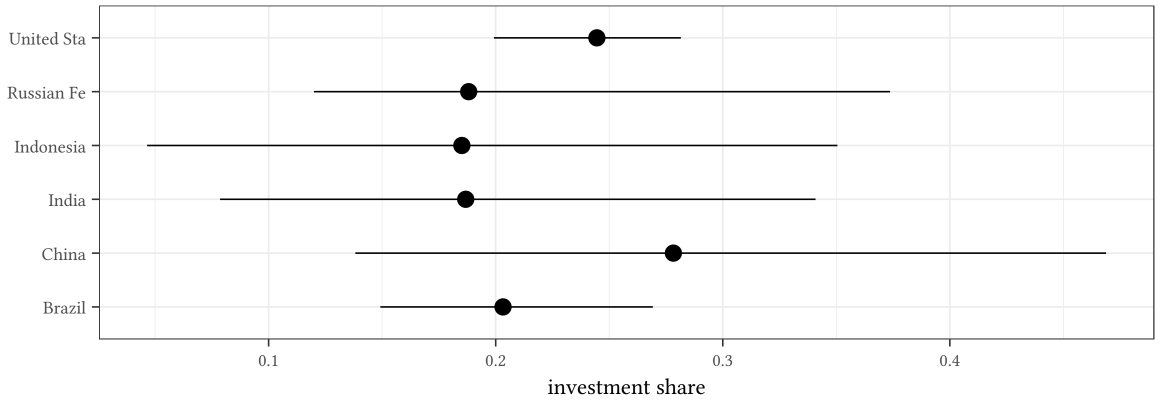
3.1 Segment plots

Sometimes we plot segments. Here we plot a range of the minimum investment share to the maximum investment share.

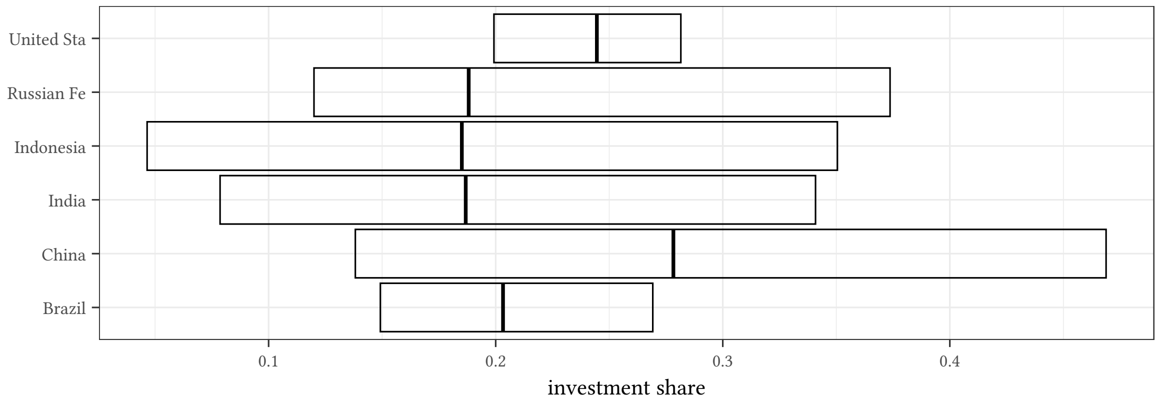
```
pwtYC(99,6) %>% group_by(country) %>%
    summarise(min=min(csh_i,na.rm=TRUE),
              max=max(csh_i,na.rm=TRUE),
              mean=mean(csh_i,na.rm=TRUE)) %>%
ggplot(aes(y=country,xmin=min,xmax=max,x=mean)) +
    geom_errorbar() + geom_point() +
    labs(x="investment share",y=NULL)
```



```
ggplot(pwtYC(99,6),aes(y=country,x=csh_i)) +
    stat_summary(fun.min=min,fun.max=max,fun=mean) +
    labs(x="investment share",y=NULL)
```



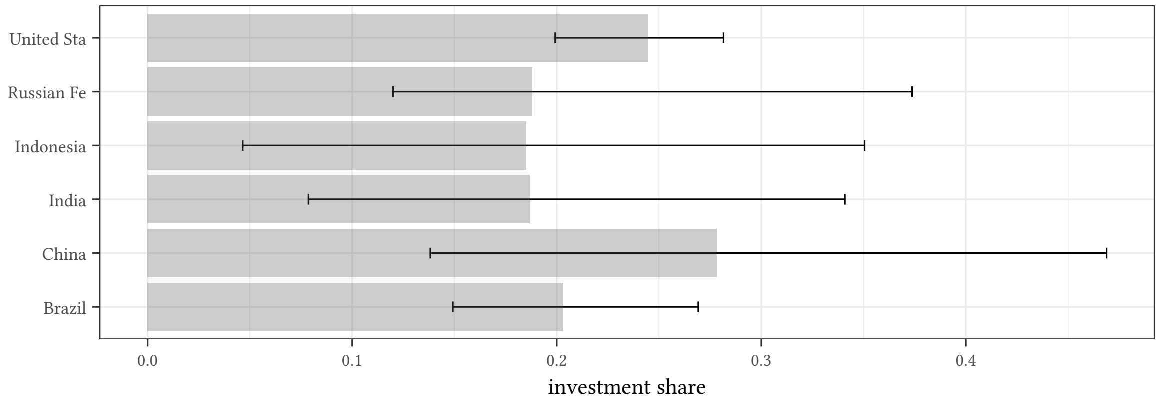
```
ggplot(pwtYC(99,6),aes(y=country,x=csh_i)) +
    stat_summary(fun.min=min,fun.max=max,fun=mean,
                 geom="crossbar") +
    labs(x="investment share",y=NULL)
```



Please don't do the following:

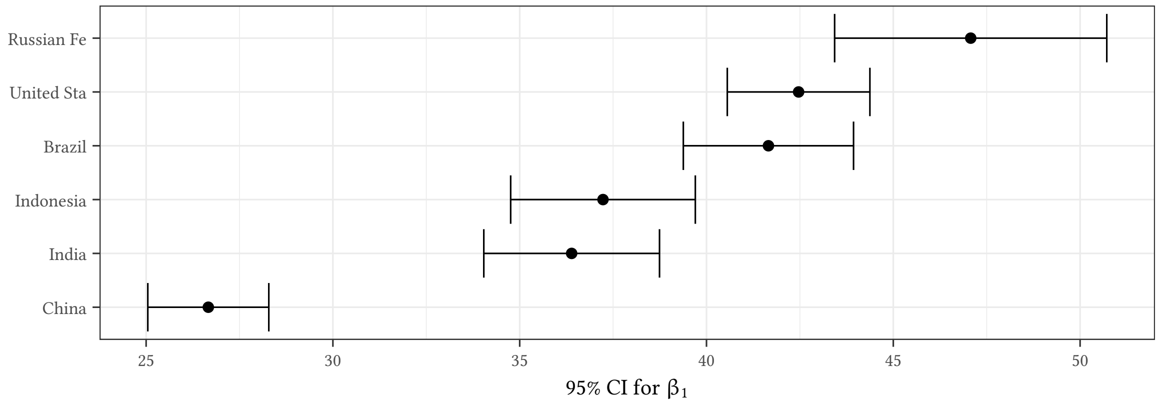
```
ggplot(pwtYC(99,6),aes(y=country,x=csh_i)) +
    stat_summary(fun.min=min,fun.max=max,
                 geom="errorbar",width=.2) +
    stat_summary(fun=mean,geom="bar",alpha=.3) +
    labs(x="investment share",y=NULL)
```

The "bar" suggests that the elements of the bar have a meaning. This might sometimes make sense, for example if the bar stands for something you can count. Most of the time bars make no sense.



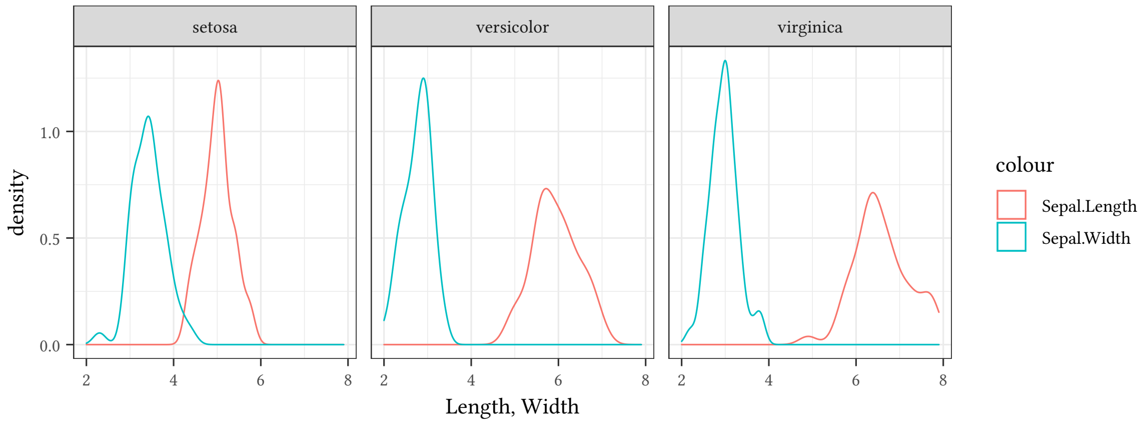
Segment plots and regression results We can also use segment plots to show regression results. In the following example we use the pwt6.3 dataset to study the relation between openc and gdpc per country:

```
reg <- lm(log(gdp) ~ csh_i:country - 1,
          data=pwtYC(99,6))
reg.ci<-data.frame(cbind(coef(reg),confint(reg)))
names(reg.ci)<-c("coef","lower","upper")
reg.ci[["country"]] <-
    factor(sub("csh_i:country","",rownames(reg.ci)))
reg.ci<-within(reg.ci,
               country<-reorder(country,coef))
ggplot(data=reg.ci,aes(y=country,x=coef)) +
    geom_point() +
    geom_errorbar(aes(xmin=lower,xmax=upper)) +
    labs(x="95\\% CI for $\\beta_1$",y=NULL)
```



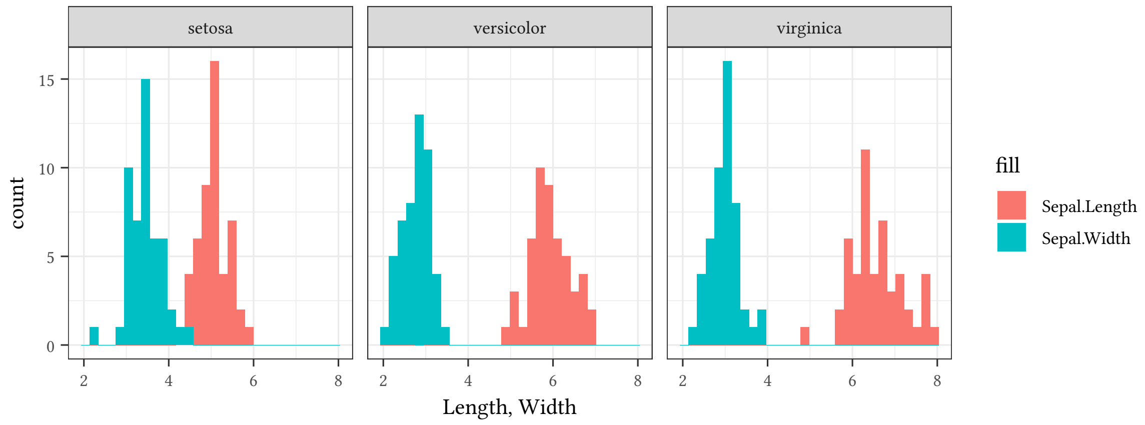
3.2 Densityplots

```
ggplot(iris) +
    geom_density(aes(x=Sepal.Length,color="Sepal.Length")) +
    geom_density(aes(x=Sepal.Width,color="Sepal.Width")) +
    facet_wrap(~ Species) + labs(x="Length, Width")
```



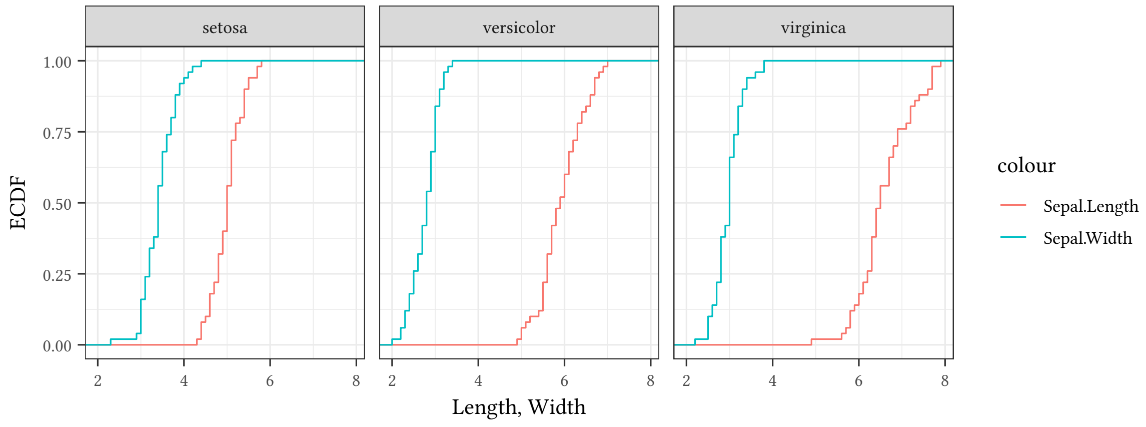
3.3 Histograms

```
ggplot(iris) +
    geom_histogram(aes(x=Sepal.Length,fill="Sepal.Length")) +
    geom_histogram(aes(x=Sepal.Width,fill="Sepal.Width")) +
    facet_wrap(~ Species) + labs(x="Length, Width")
```



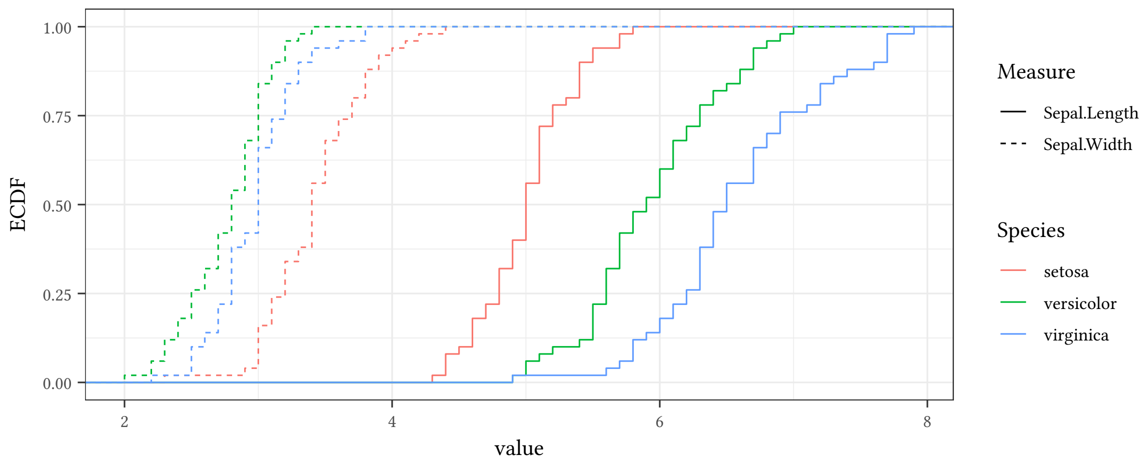
3.4 Empirical cumulative distribution

```
ggplot(iris) +
    stat_ecdf(aes(x=Sepal.Length,color="Sepal.Length")) +
    stat_ecdf(aes(x=Sepal.Width,color="Sepal.Width")) +
    facet_wrap(~ Species) + labs(x="Length, Width", y="ECDF")
```



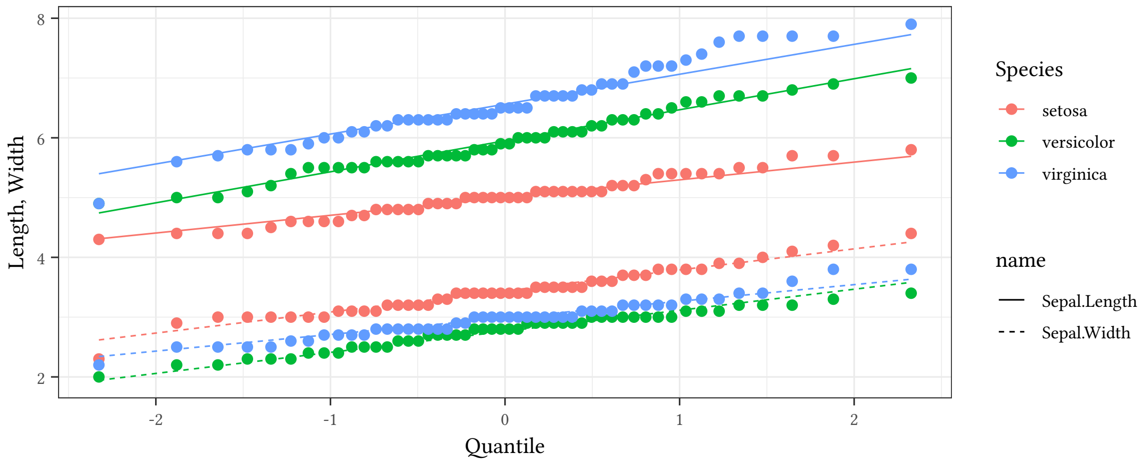
```
iris %>% slice_head(n=3)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa

iris %>% tidyr::pivot_longer(cols=starts_with("Sepal.")) %>%
    slice_head(n=3)

# A tibble: 3 x 5
  Petal.Length Petal.Width Species name         value
         <dbl>       <dbl> <fct>   <chr>        <dbl>
1          1.4         0.2 setosa  Sepal.Length   5.1
2          1.4         0.2 setosa  Sepal.Width    3.5
3          1.4         0.2 setosa  Sepal.Length   4.9
```

```
tidyr::pivot_longer(iris,cols=starts_with("Sepal.")) %>%
    ggplot(aes(x=value,color=Species,lty=name)) +
    stat_ecdf() + labs(lty="Measure",y="ECDF")
```



3.5 Q-Q plots

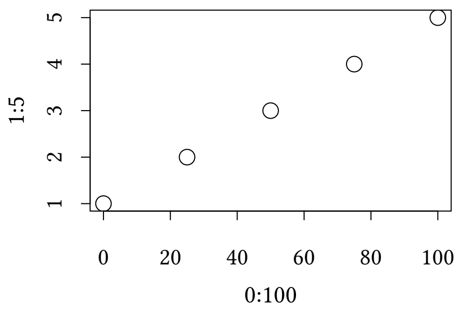
```
tidyr::pivot_longer(iris,cols=starts_with("Sepal.")) %>%
    ggplot(aes(sample=value,color=Species,lty=name)) +
    stat_qq() + stat_qq_line()  + labs(y="Length, Width", x="Quantile")
```



3.6 Sample Q-Q plots

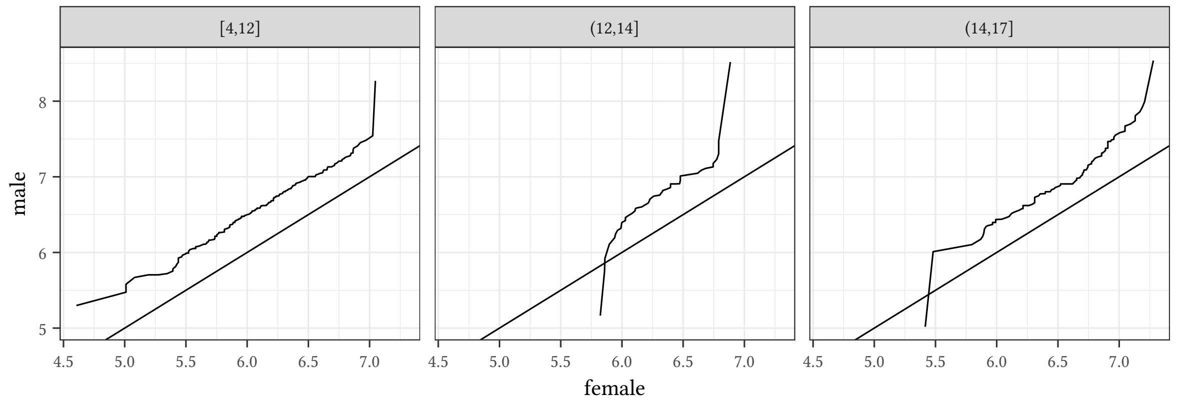
An example:

```
data.frame(qqplot(0:100,1:5))

    x y
1   0 1
2  25 2
3  50 3
4  75 4
5 100 5
```



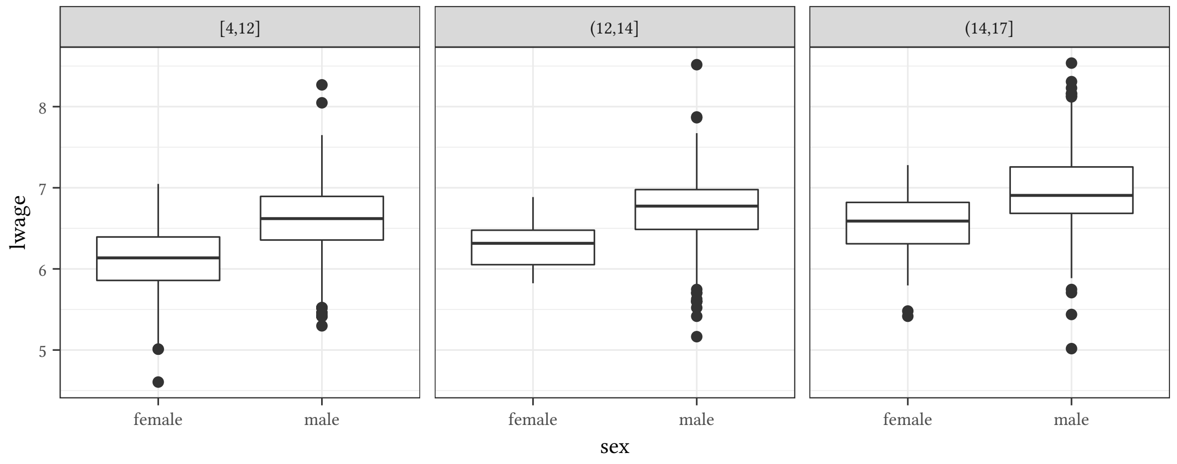
We use the `qqplot` function to prepare data for `ggplot`:

```
data(Wages,package="Ecdat")
Wages %>% mutate(edG = cut_number(ed,n=3)) %>%
    group_by(edG) %>%
    summarise(data.frame(qqplot(plot.it=FALSE,lwage[sex!="male"],lwage[sex=="male"]))) %>%
    ggplot(aes(x=x,y=y)) + geom_line() + labs(x="female",y="male") +
    geom_abline(slope=1,intercept=0) + facet_wrap(~edG)
```



3.7 Boxplots

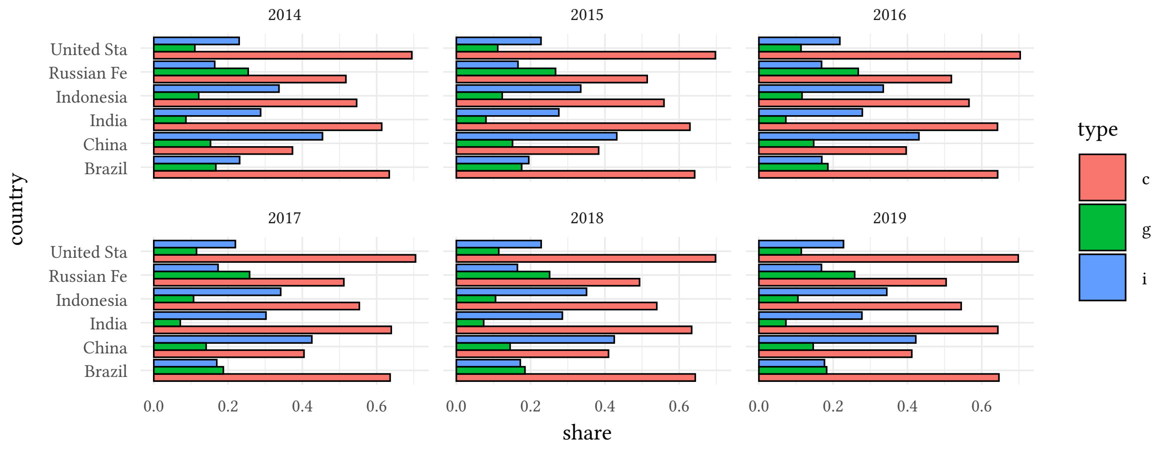
```
Wages %>% mutate(edG = cut_number(ed,n=3)) %>%
    ggplot(aes(y=lwage,x=sex)) + geom_boxplot() + facet_wrap(~edG)
```



3.8 Barcharts

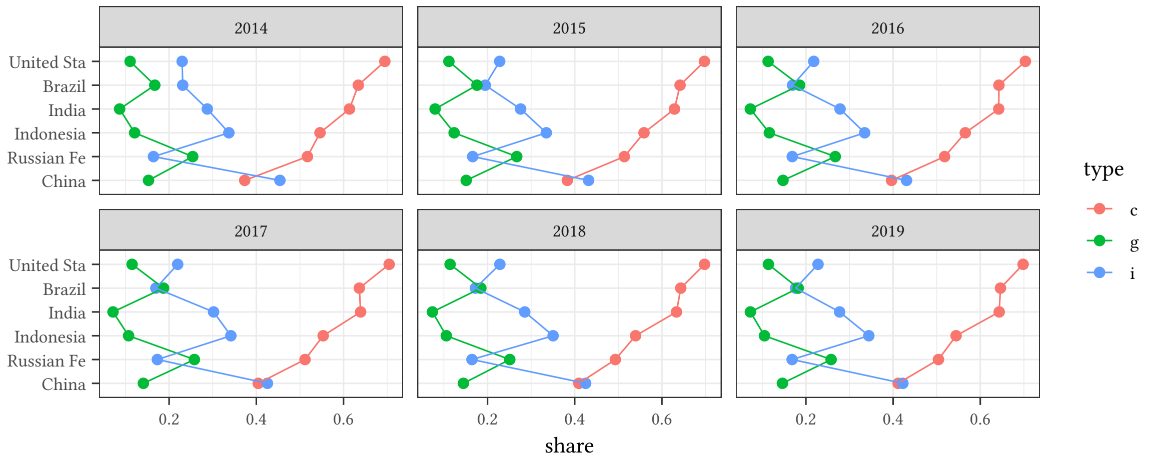
ggplot can also do bar charts:

```
pwtLong %>% filter(year > max(year)-6) %>%
ggplot(aes(y=country, x=share, fill=type)) +
geom_bar(stat="identity", color="black", position=position_dodge())+
  theme_minimal() + facet_wrap(~year)
```



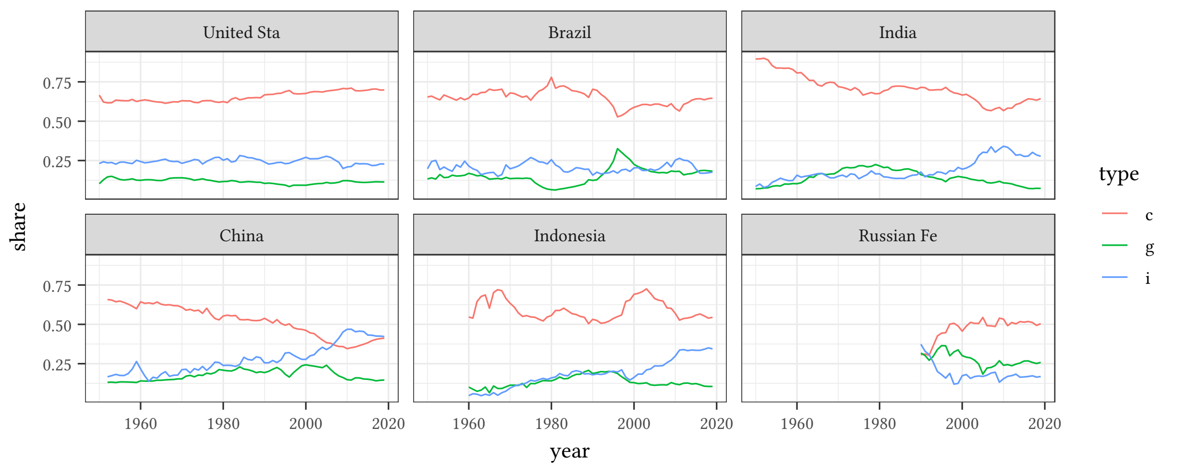
We should note that often a `dotplot` or `xyplot` presents the same data in a better way.

```
pwtLong %>% filter(year > max(year)-6) %>%
ggplot(aes(x=reorder(country,share,max), y=share, group=type, color=type)) +
    geom_point() + geom_line() + facet_wrap(~year) + coord_flip() + labs(x=NULL)
```



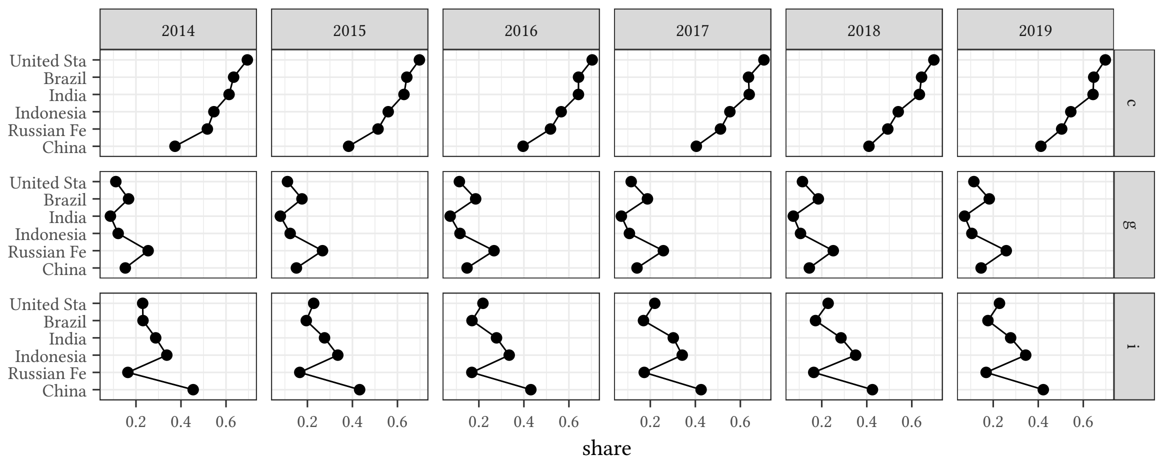
(`coord_flip`, so that lines are drawn properly).

```
pwtLong %>% ggplot(aes(x=year,y=share,color=type)) +
    facet_wrap(~reorder(country,share,max)) +
    geom_line()
```



3.9 Coplots

```
pwtLong %>% filter(year > max(year)-6) %>%
ggplot(aes(x=reorder(country,share,max), y=share, group=1)) +
    geom_point() + geom_line() + facet_grid(type~year) + coord_flip() + labs(x=NULL)
```

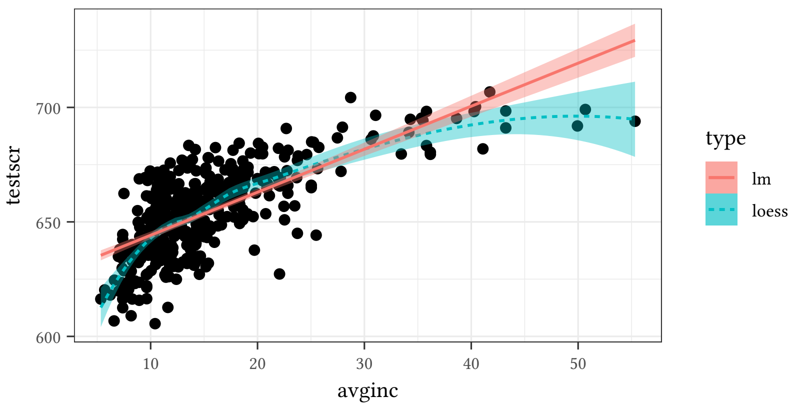


3.10 Parameters

3.10.1 Types of lines

With `lattice` we would choose between different types of lines with `type`. With ggplot we use different geoms. In the following graph we use `aes(color=...)` to create a legend for the different geoms.

```
data(Caschool,package="Ecdat")
ggplot(data=Caschool,aes(x=avginc,y=testscr))+geom_point()+
    geom_smooth(aes(color="loess",fill="loess",lty="loess"))+
    geom_smooth(aes(color="lm",fill="lm",lty="lm"),method="lm") +
    labs(fill="type",color="type",lty="type")
```

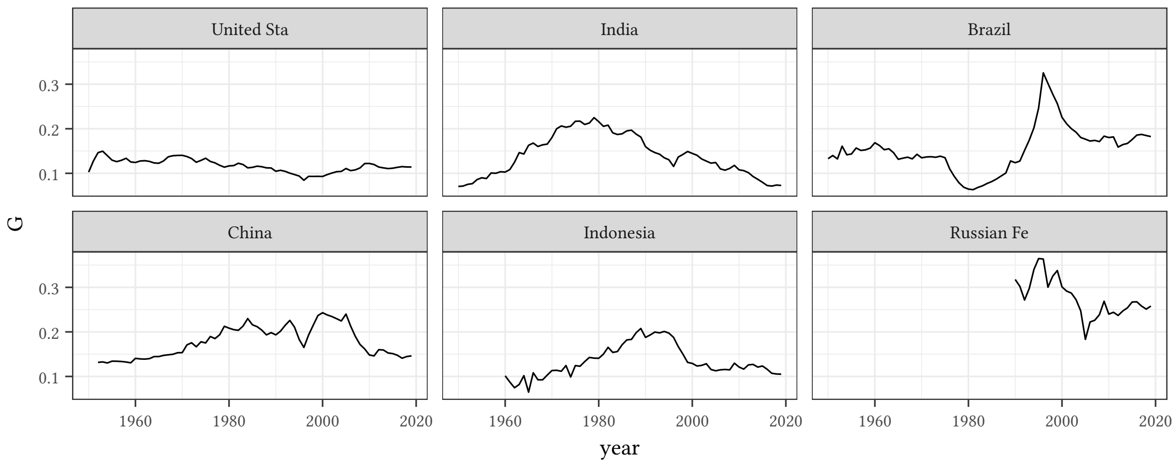


3.10.2 Axes

Different scales for different panels As with `lattice`, also `ggplot` chooses the same scale for all panels in a plot. This can be changed with the help of the parameter `scales` in `facet_wrap`.

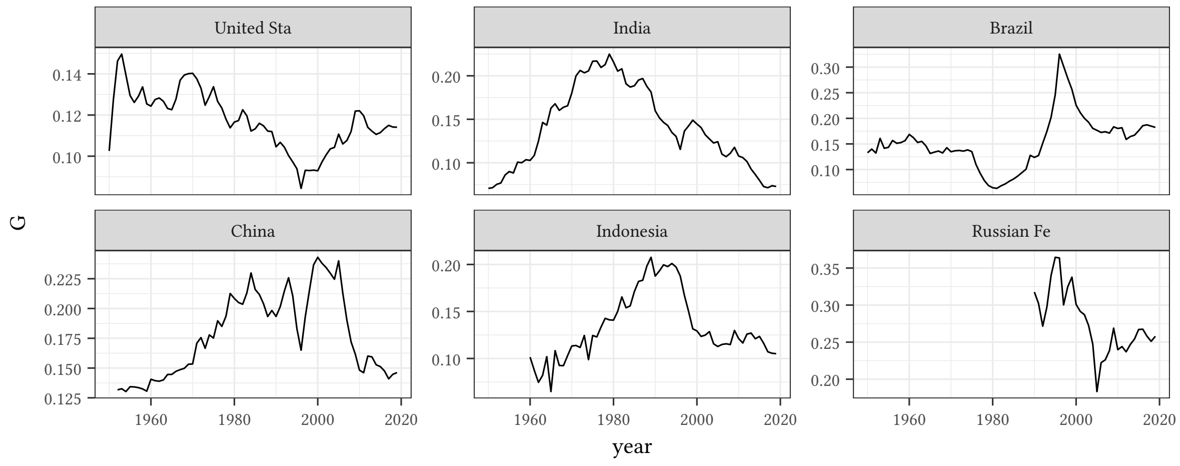
Same scale (the default):

```
ggplot(pwtYC(99,6), aes(x=year,y=csh_g)) +
    geom_line() +
    facet_wrap(~reorder(country,csh_g)) +
    labs(y="G")
```



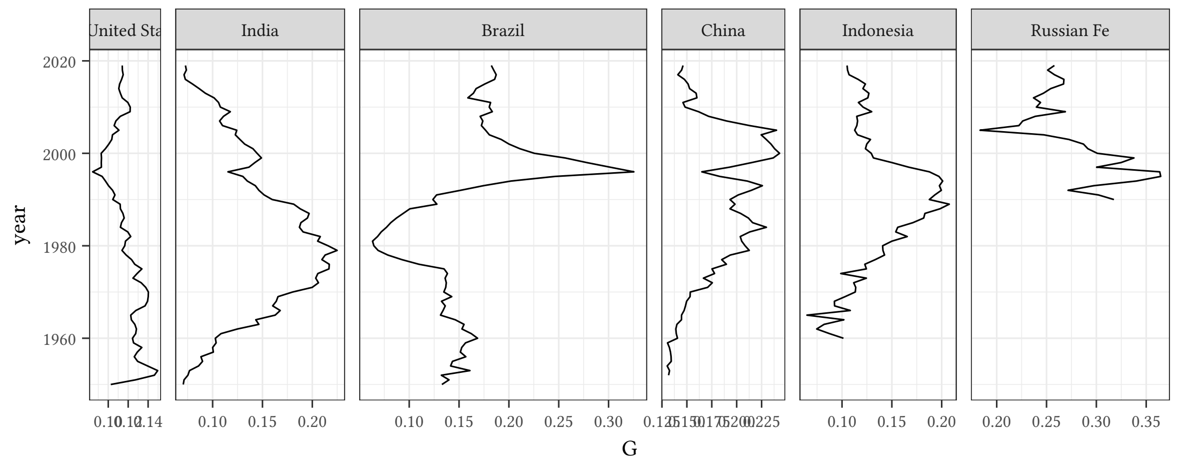
Free scale (`scales=list(x="same",y="free")`):

```
ggplot(pwtYC(99,6), aes(x=year,y=csh_g)) +
    geom_line() +
    facet_wrap(~reorder(country,csh_g),scales="free_y") +
    labs(y="G")
```



Sliced scale (`facet_grid(...,space='free')`, scales have the same scale, but different origin (this is different than in `lattice`):

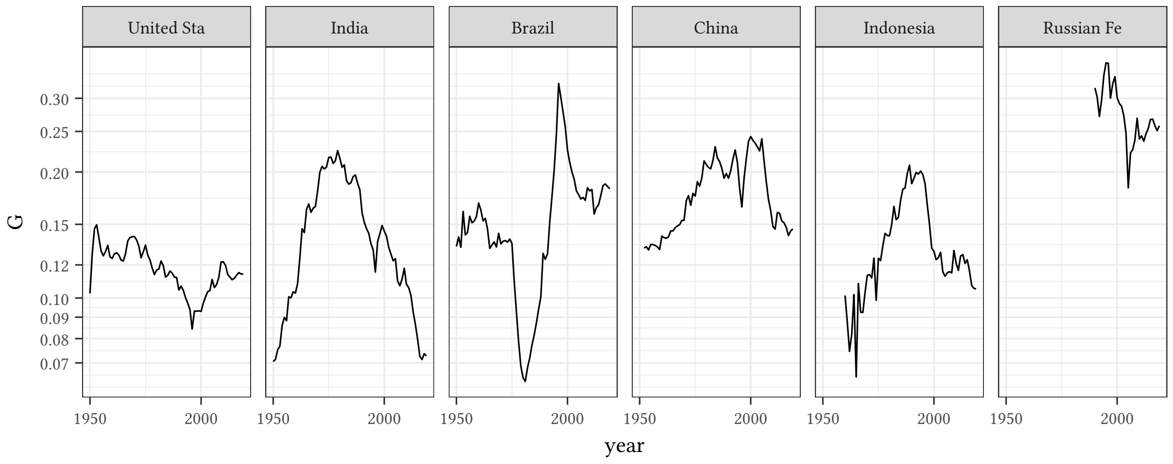
```
ggplot(pwtYC(99,6), aes(x=year,y=csh_g)) +
    geom_line() +  facet_grid(.~reorder(country,csh_g),scales="free",space="free") +
    labs(y="G") + coord_flip()
```



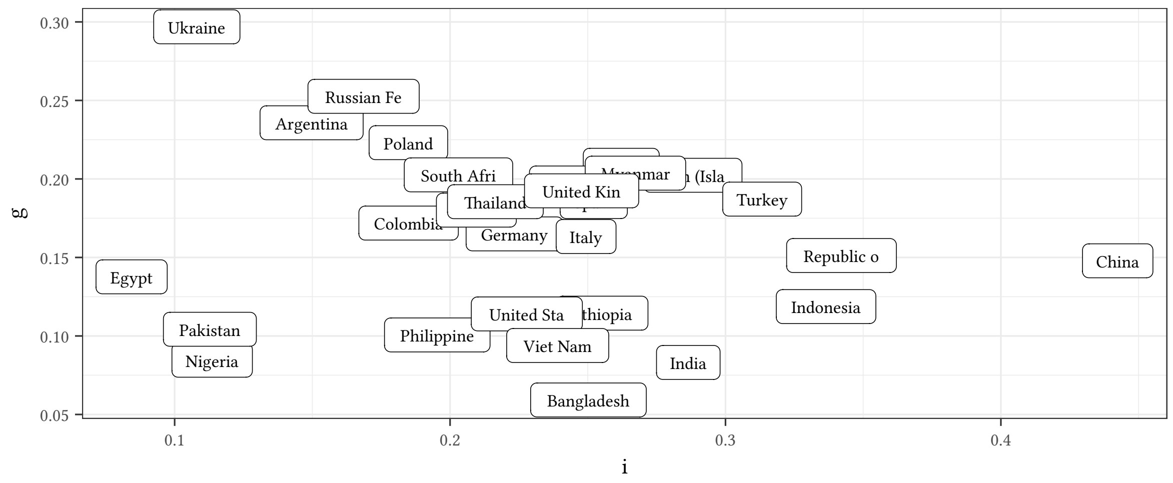
Individual axes

We can influence *where* an axis is labelled as follows:

```
ggplot(data=pwtYC(99,6), aes(x=year,y=csh_g)) +
    geom_line() + facet_grid(.~reorder(country,csh_g)) + labs(y="G") +
    scale_y_log10(breaks=c(.07,.08,.09,.1,.12,.15,.2,.25,.3)) +
    scale_x_continuous(breaks=c(1950,2000))
```



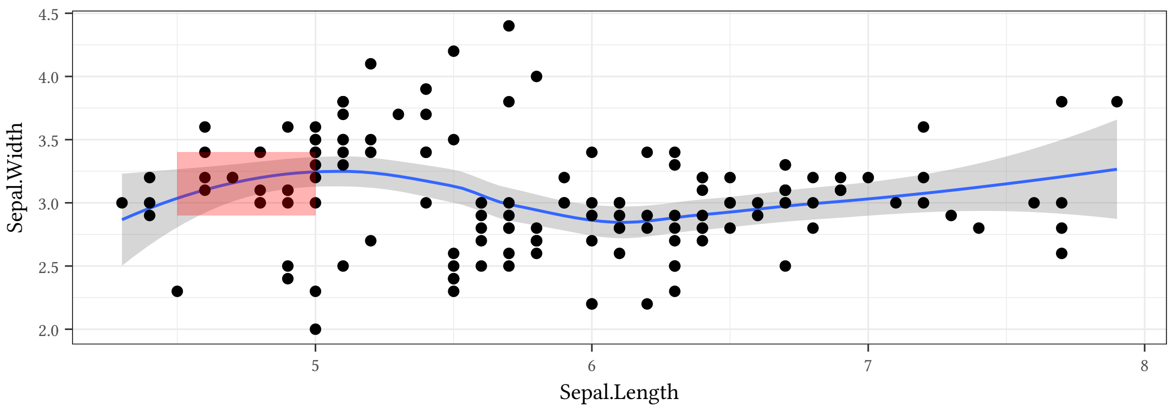
```
pwtYC(10,30) %>% group_by(country) %>%
    summarise(i=median(csh_i,na.rm=TRUE),g=median(csh_g,na.rm=TRUE)) %>%
    ggplot(aes(x=i,y=g,label=country)) + geom_label(size=3)
```



3.11 Zooming

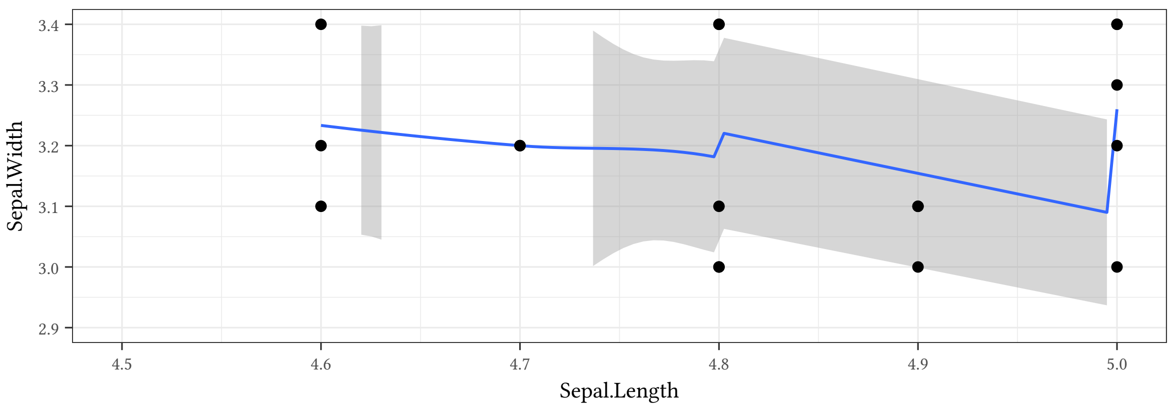
Sometimes we want to show only part of the data. No problem if the graph shows nothing but the data. If, however, the graph only shows statistics, e.g., a smooth line, the shape of the line depends on the data that is included.

```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_smooth() + geom_point() +
    annotate("rect",xmin=4.5,xmax=5,ymin=2.9,
             ymax=3.4,alpha=.3,fill="red")
```



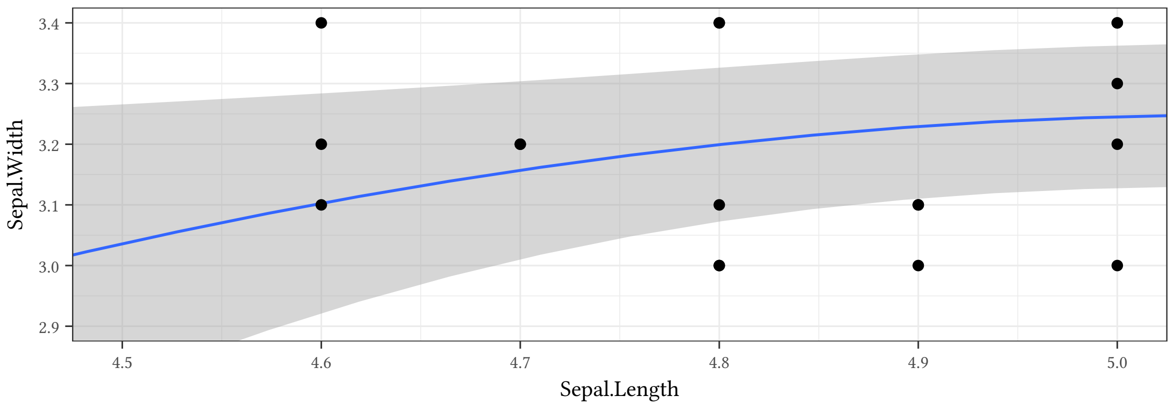
The following graph uses only a subset of the data to calculate the smooth line.

```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_smooth() + geom_point() +
    xlim(c(4.5,5)) + ylim(c(2.9,3.4))
#
```



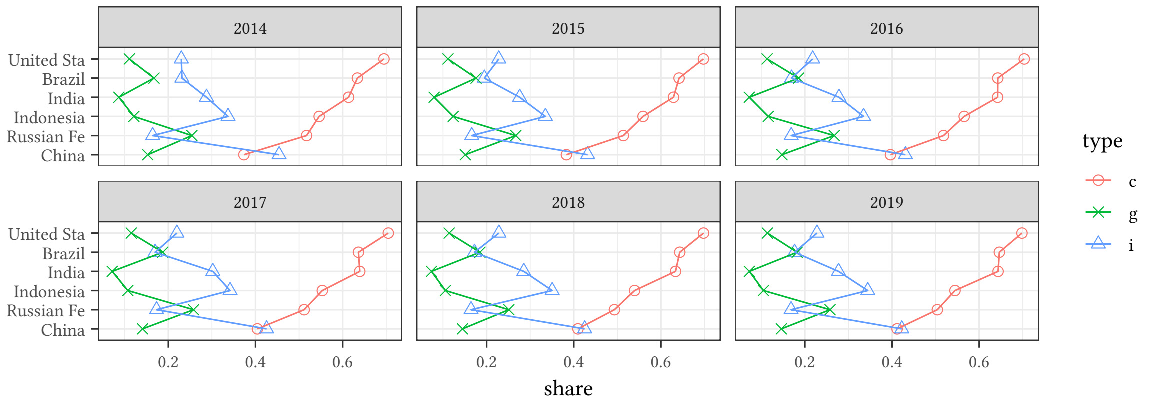
The following graph uses the entire data to calculate the smooth line.

```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_smooth() + geom_point() +
    coord_cartesian(xlim=c(4.5,5),ylim=c(2.9,3.4))
#
```

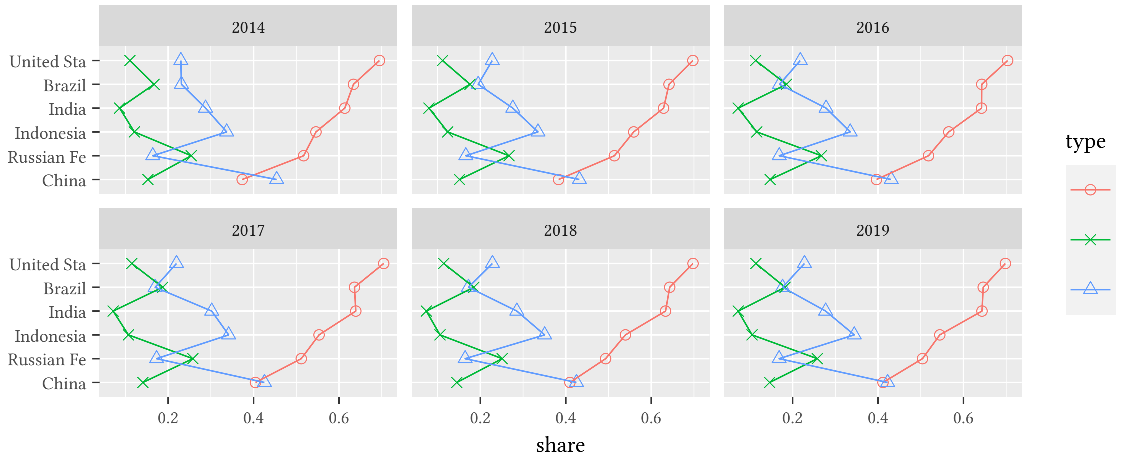


3.12 Themes

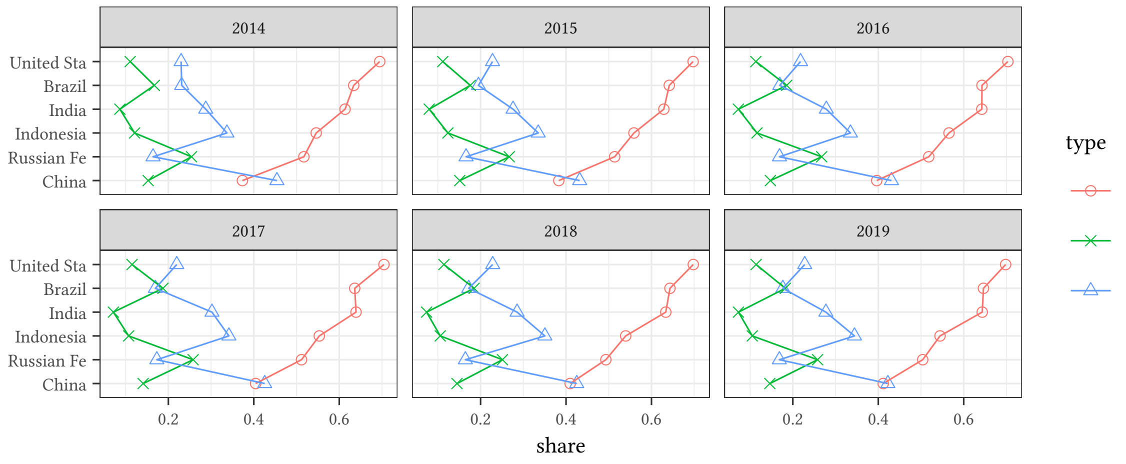
```
pwtLong %>% filter(year > max(year)-6) %>%
    ggplot(aes(x=reorder(country,share,max), y=share, group=type, shape=type, color=type)) +
    labs(x=NULL) +
    geom_point() + geom_line() + facet_wrap(~year) + coord_flip() -> p
p
```



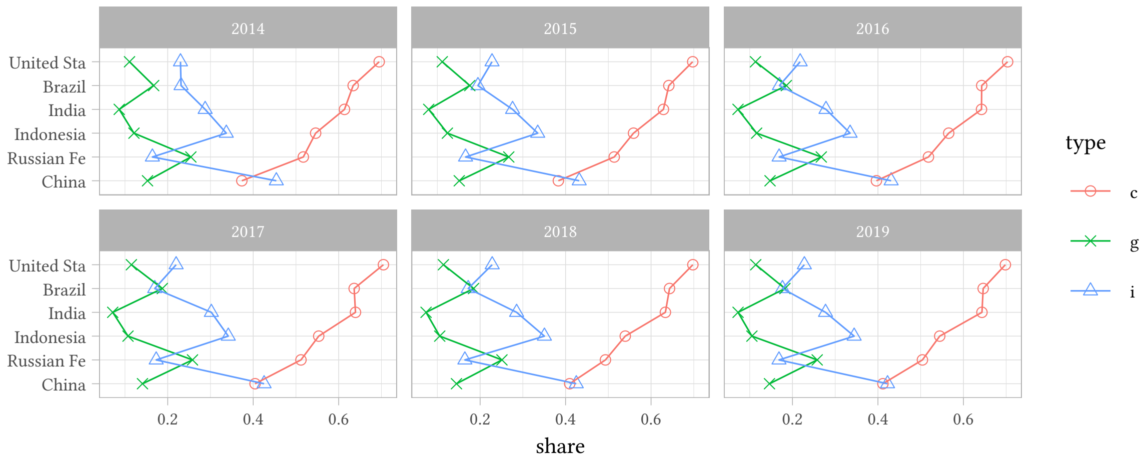
```
p + theme_gray()
```



```r
p + theme_bw()
```

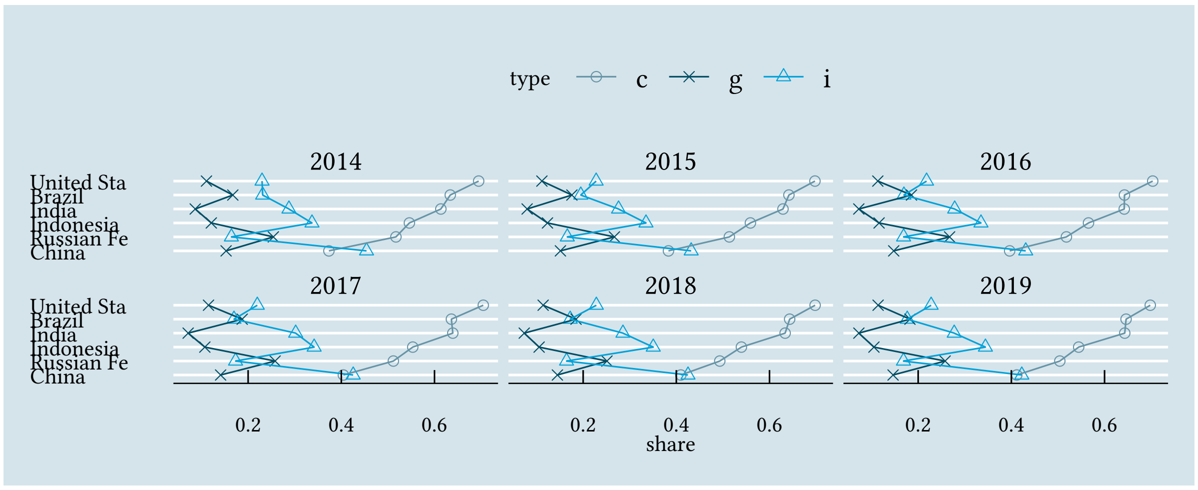


```r
p + theme_light()
```

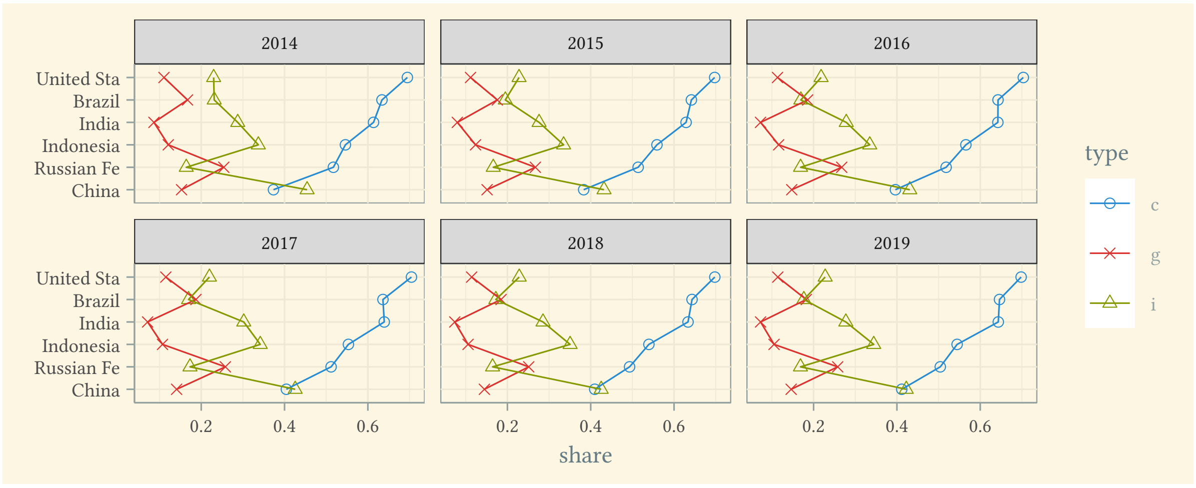


ggthemes The `ggthemes` library offers a number of additional themes.

```
library(ggthemes)
p + theme_economist() + scale_colour_economist()
```



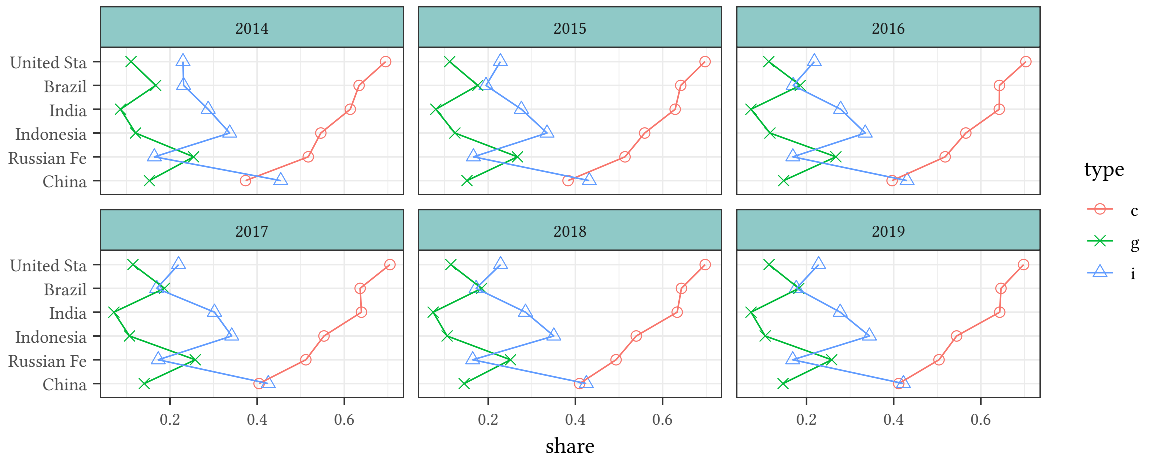
```
library(ggthemes)
p + theme_solarized() + scale_colour_solarized("blue")
```



g

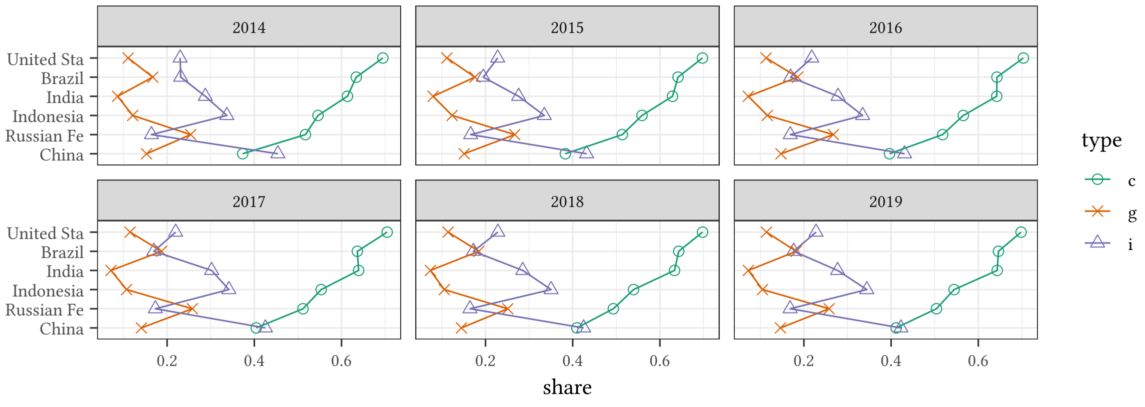
Setting the strip to a specific color

```
p + theme(strip.background=element_rect(fill="#8FC9C7"))
```



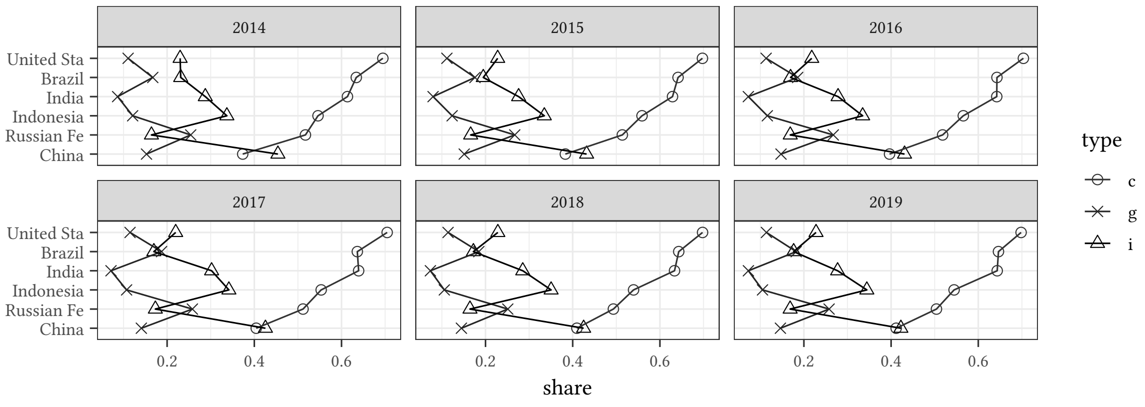
Setting colors Parts of the plot which do not represent data can be influenced with `theme_set()`. If we also want to change the presentation of the data once and for all, we can redefine the `ggplot` function:

```
ggplot <- function(...) ggplot2::ggplot(...) +
                            scale_fill_brewer(palette="Dark2") +
                            scale_color_brewer(palette="Dark2") +
                            scale_shape_manual(values=c(1,4,2,3,0,5:10))
```



Lines will be drawn in a different color. Points will have a different shape.
If no colors are desired, then use `scale_color_grey(end=0)` and `scale_fill_grey(end=0)`

```
ggplot <- function(...) ggplot2::ggplot(...) +
                                scale_fill_grey(end=0) +
                                scale_color_grey(end=0) +
                                scale_shape_manual(values=c(1,4,2,3,0,5:10))
```



4 Nominal data

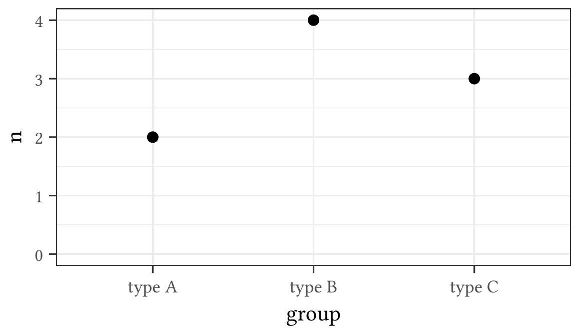
The case of purely nominal data is rare. However, sometimes we want to present a simplified version (where only nominal categories matter) of a richer dataset in the description.

4.1 Nominal univariate

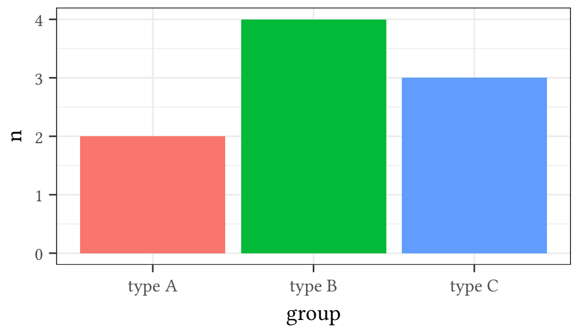
```
set.seed(123)
nomD <- data.frame(n = rbinom(3,10,.3),
                   group = c("type A","type B","type C"))
nomD
```

```
  n  group
1 2 type A
2 4 type B
3 3 type C
```

```
ggplot(nomD,aes(x=group,y=n)) +
    geom_point() + expand_limits(y=0)
#
```

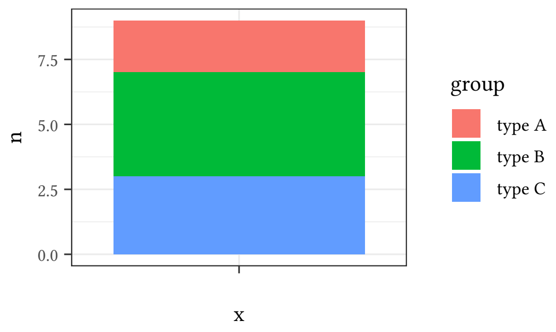


```
ggplot(nomD,aes(x=group,y=n,fill=group)) +
    geom_bar(stat="identity") +
    theme(legend.position="none")
```



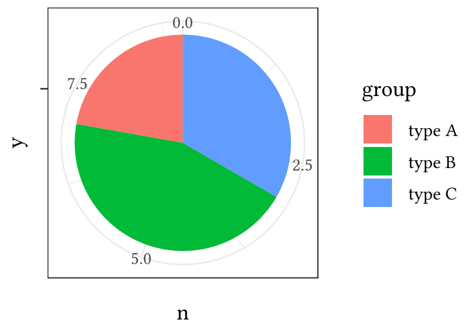
waste of ink

```
ggplot(nomD,aes(x="",y=n,fill=group)) +
    geom_bar(stat="identity")
#
```



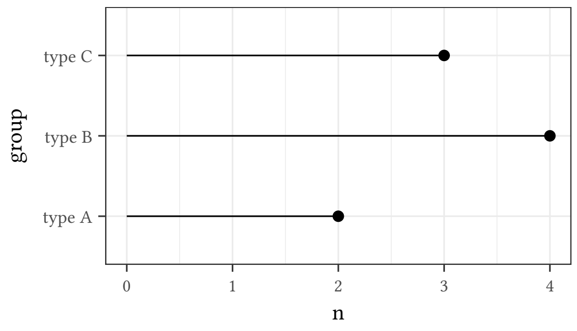
waste of ink, categories hard to compare

```
ggplot(nomD,aes(x=n,y="",fill=group)) +
    geom_bar(stat="identity") +
    coord_polar()
```



waste of ink, categories very hard to compare

```
ggplot(nomD,aes(x=n,y=group)) +
    geom_point() +
    geom_segment(aes(x=0,xend=n,yend=group)) +
    expand_limits(x=0)
```



Juxtaposed bar charts

- waste of ink

Stacked bar charts

- waste of ink
- harder to compare values

Pie chart

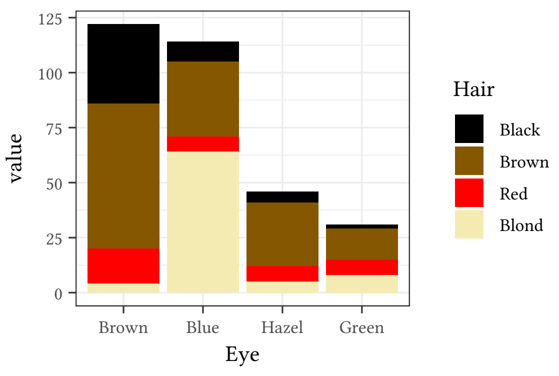
- - The eye is not good at comparing angles (except 90° and 180°).
  Avoid pie charts (unless 90° and 180° are of special significance).

4.2 Nominal bivariate

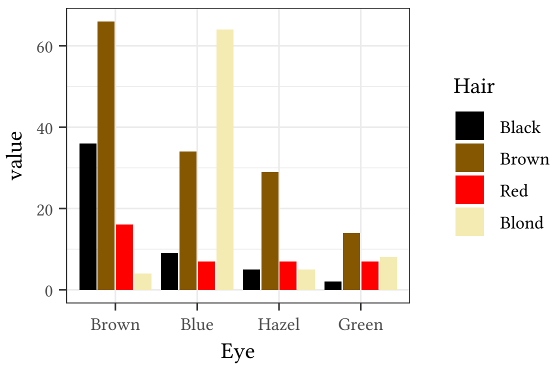
- Barplots
- Bubbleplots
- Mosaicplots (Hartigan and Kleiner, 1984)
- Dot-plots

```
data(HairEyeColor)
reshape2::melt(HairEyeColor)

    Hair   Eye  Sex value
1  Black Brown Male    32
2  Brown Brown Male    53
3    Red Brown Male    10
4  Blond Brown Male     3
5  Black  Blue Male    11
6  Brown  Blue Male    50
7    Red  Blue Male    10
8  Blond  Blue Male    30
9  Black Hazel Male    10
10 Brown Hazel Male    25
 [ reached 'max' / getOption("max.print") -- omitted 22 rows ]

myColor <- c("black","#855700","red","#f4ebb3")
myFillSc <- scale_fill_manual(values=myColor)
myColSc <- scale_color_manual(values=myColor)
```

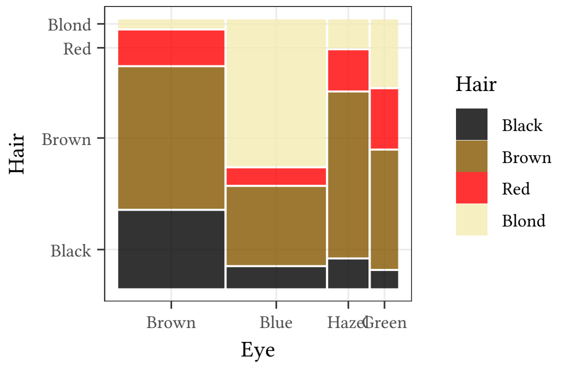
```
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>%
    ggplot(aes(x=Eye,y=value,fill=Hair)) +
    geom_bar(stat='identity') + myFillSc
```



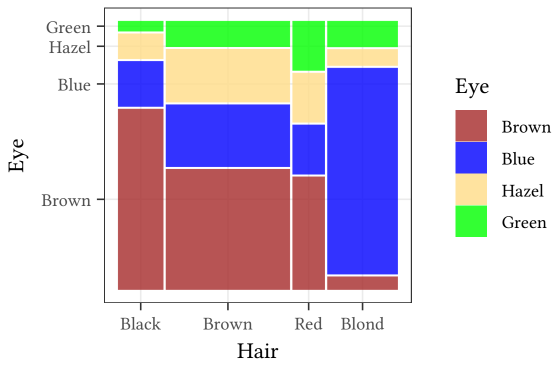
```
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>%
    ggplot(aes(x=Eye,y=value,fill=Hair)) +
    geom_bar(stat='identity',position="dodge2") +
    myFillSc
```



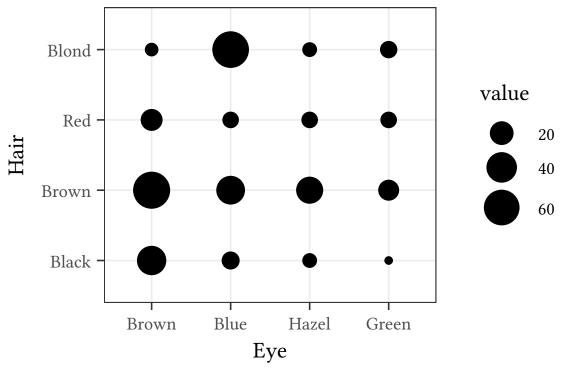
```
library(ggmosaic)
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>% ggplot() +
    geom_mosaic(aes(x=product(Hair,Eye),
                    weight=value,fill=Hair)) +
    myFillSc
```



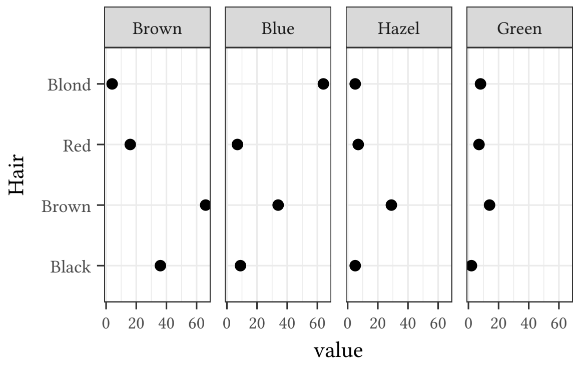
```
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>% ggplot() +
    geom_mosaic(aes(x=product(Eye,Hair),
                    weight=value,fill=Eye)) +
    scale_fill_manual(values=c("brown","blue",
                               "#ffdd88","green"))
```



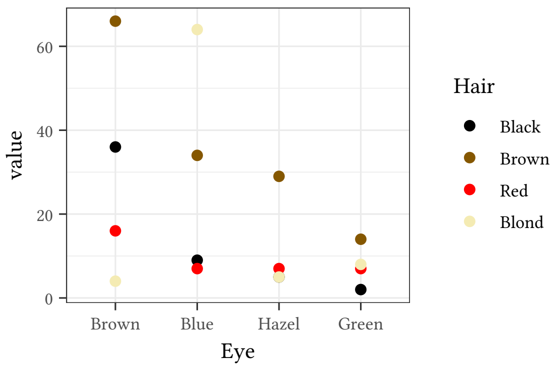
```
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>%
    ggplot(aes(x=Eye,y=Hair,size=value)) +
    geom_point()
```



```
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>%
    ggplot(aes(y=Hair,x=value)) +
    geom_point() + facet_wrap(vars(Eye),nrow=1)
```



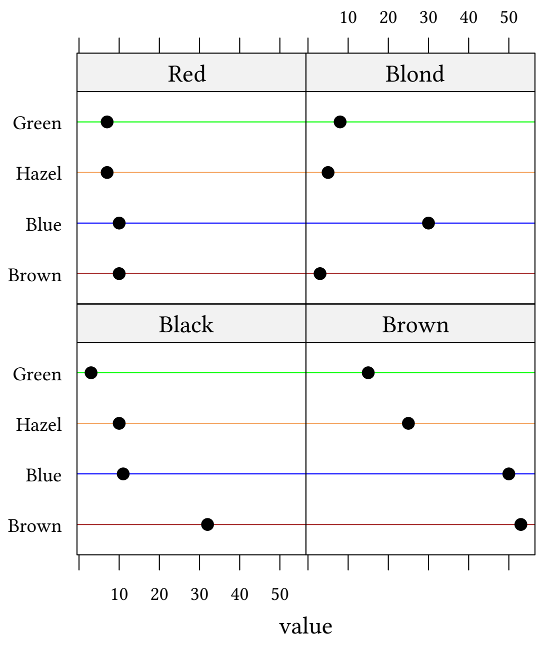
```
reshape2::melt(HairEyeColor) %>%
    filter(Sex=="Female") %>%
    ggplot(aes(x=Eye,y=value,color=Hair)) +
    geom_point() + myColSc
```



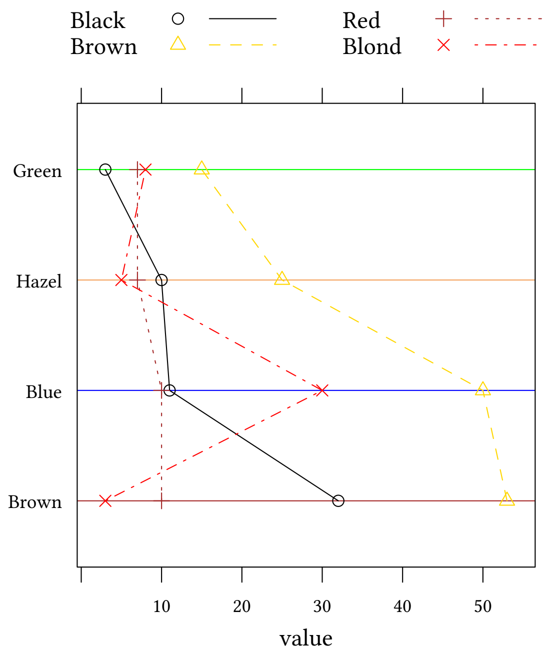
(Spineplots are very similar to mosaicplots)

Multiway dot-plots Multiway dot-plots are another possibility to present two way count data:

```
HairEyeMale <- reshape2::melt(HairEyeColor[,,"Male"])
colEye<-c("brown","blue","sandybrown","green")
myTheme<-within(lTheme,dot.line$col<-colEye)
(d1<-dotplot(Eye ~ value | Hair,data=HairEyeMale,par.settings=myTheme))
```



```
colHair<-c("black","gold","brown","red")
myTheme<-within(lTheme,{
    dot.line$col<-colEye
    superpose.symbol$col<-colHair
    superpose.line$col<-colHair})
keys<-list(space="top",columns=2,lines=TRUE)
(d2<-dotplot(Eye ~ value ,group=Hair,data=HairEyeMale,t=c("p","a"),auto.key=keys,par.settings=m
```



Juxtaposed barplot – hard to see a structure in the sub categories

Stacked barplot + easy to assess sum of observations in each category

Mosaicplot + easy to compare relative frequencies

+ easy to assess indepence of categories

Bubbleplot - hard to compare relative frequencies

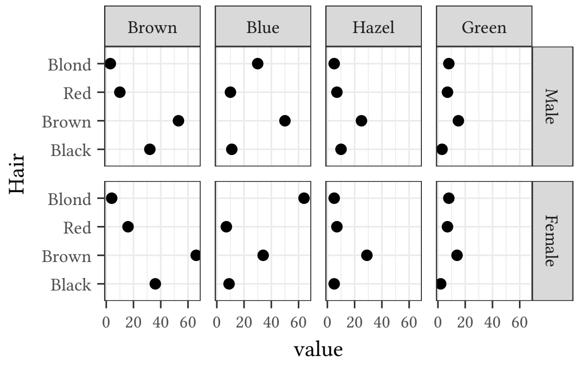
+ good if the number of categories is large (in particular for numeric categories)

Dot-plot + easy to look up and to compare absolute frequencies

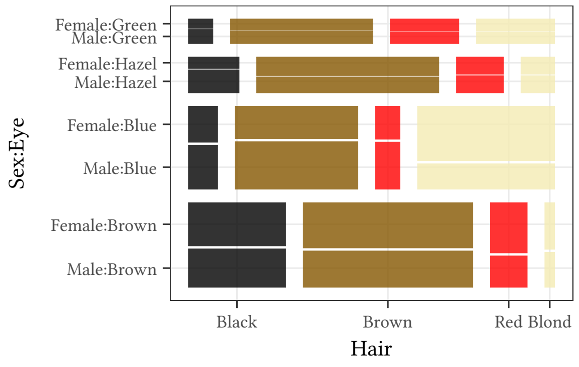
+ easy to see a pattern

4.3 Nominal multivariate

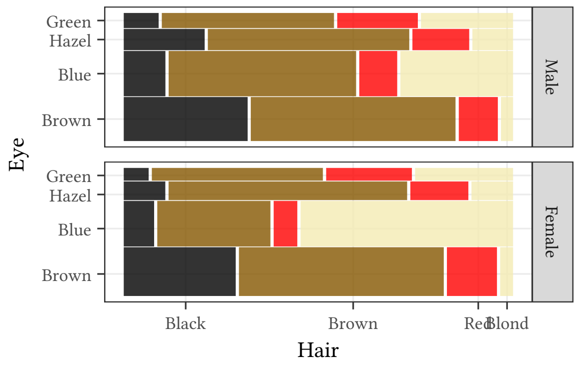
```
reshape2::melt(HairEyeColor) %>%
    ggplot(aes(y=Hair,x=value)) +
    geom_point() +
    facet_grid(cols=vars(Eye),rows=vars(Sex))
```



```
reshape2::melt(HairEyeColor) %>% ggplot() +
    geom_mosaic(aes(x=product(Sex,Hair,Eye),
                    weight=value,fill=Hair),
                divider=mosaic("v"),offset=.05,
                show.legend=FALSE) +
    myFillSc
```



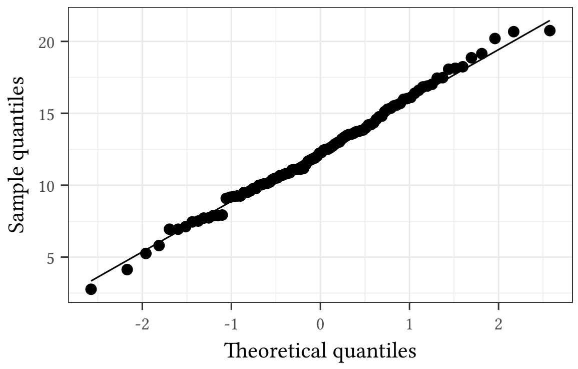
```
reshape2::melt(HairEyeColor) %>% ggplot() +
    geom_mosaic(aes(x=product(Hair,Eye),
                    weight=value,fill=Hair),
                show.legend=FALSE,
                divider=mosaic("v")) +
    myFillSc + facet_grid(vars(Sex))
```



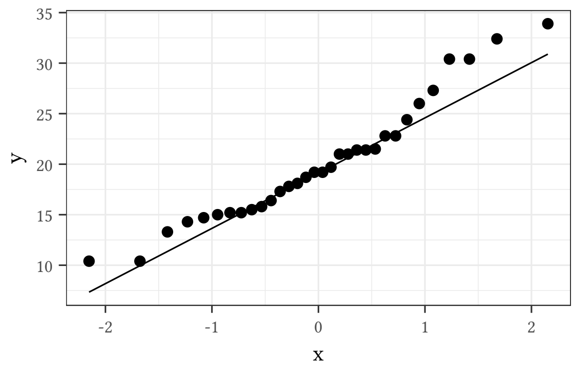
5 Continuous data – distributions

5.1 Diagnostic plots for continuous variables

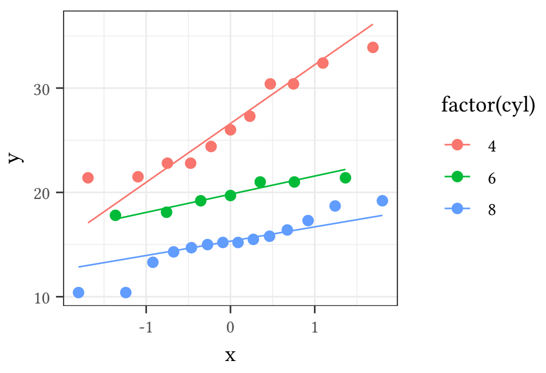
```
set.seed(123)
data.frame(x = rnorm(100,mean=12,sd=4)) %>%
    ggplot(aes(sample=x)) + stat_qq() +
    stat_qq_line() +
    labs(x="Theoretical quantiles",
         y="Sample quantiles")
```



```
mtcars %>%
    ggplot(aes(sample=mpg)) +
    stat_qq() + stat_qq_line()
```



```
mtcars %>%
    ggplot(aes(sample=mpg,color=factor(cyl))) +
    stat_qq() + stat_qq_line()
```



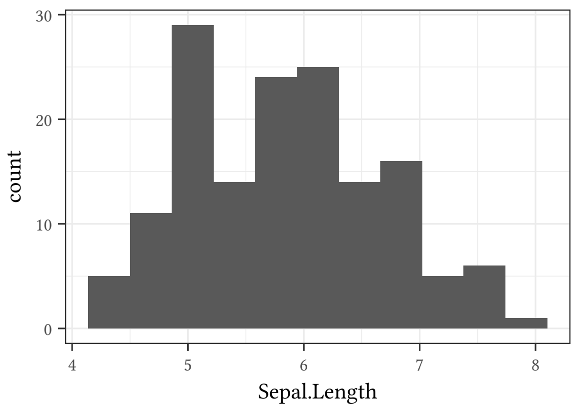
qqnorm compares with a given (theoretical) distribution. qqplot compares with a given empirical distribution.

5.2 One continuous plus one nominal

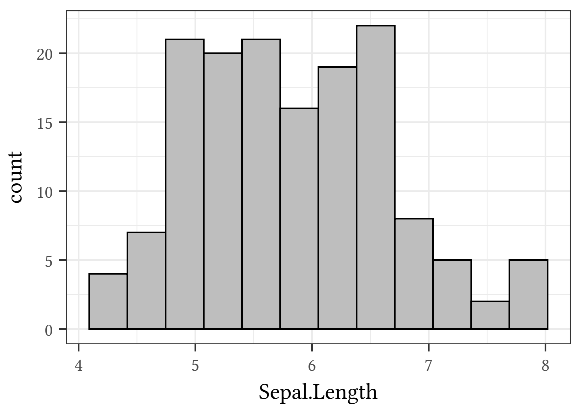
5.2.1 Histograms

Histograms don't let you see a difference:

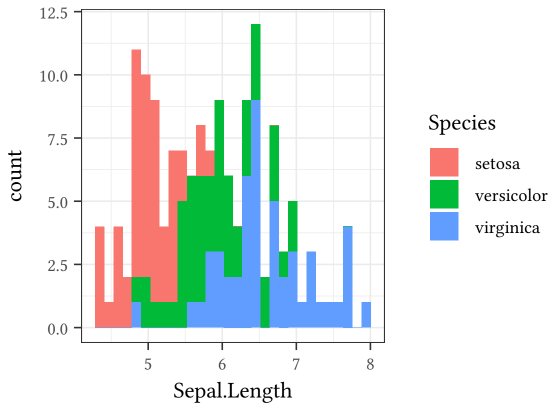
```
iris %>% ggplot(aes(Sepal.Length)) +
    geom_histogram(bins=11)
```



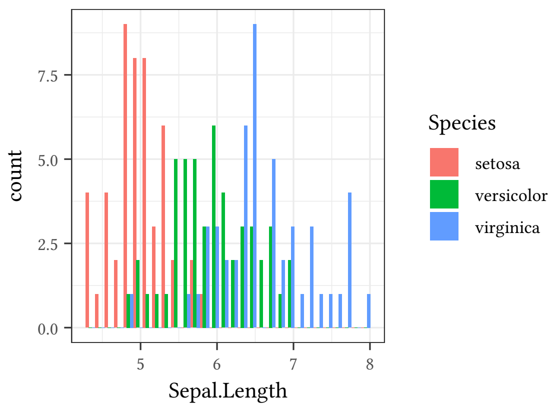
```
ggplot(iris,aes(Sepal.Length)) +
    geom_histogram(bins=12,fill="gray",color="black")
```



```
ggplot(iris,aes(x=Sepal.Length,fill=Species)) +
    geom_histogram()
```

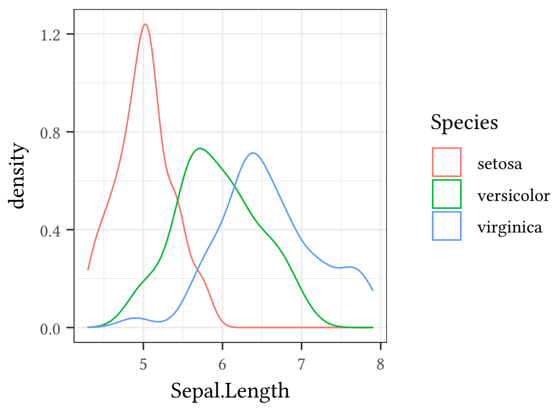


```
ggplot(iris,aes(x=Sepal.Length,fill=Species)) +
    geom_histogram(position="dodge")
```

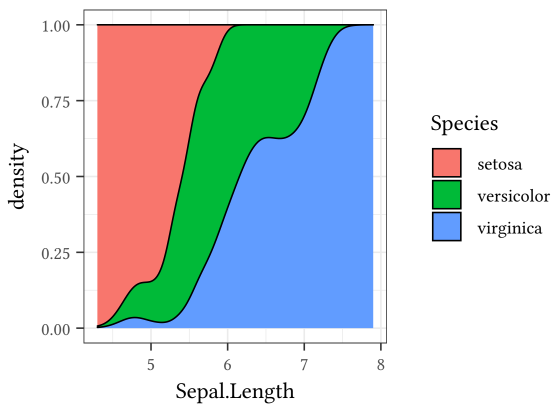


5.2.2 Densities and conditional densities

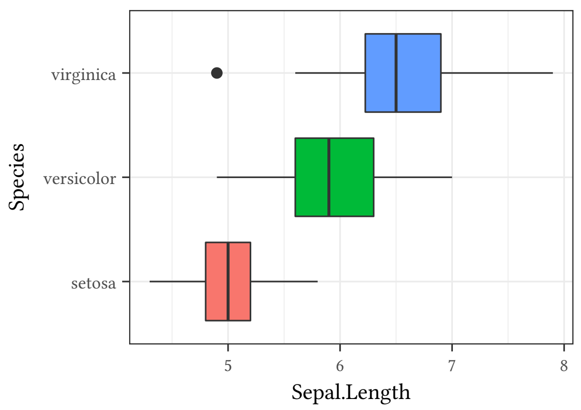
```
ggplot(iris,aes(x=Sepal.Length,color=Species)) +
    geom_density()
```



```
ggplot(iris, aes(Sepal.Length,fill = Species))+
    geom_density(position = "fill")
```



```
ggplot(iris, aes(Sepal.Length, y=Species, fill=Species)) +
    geom_boxplot() + theme(legend.position="none")
```



Elements of the boxplot

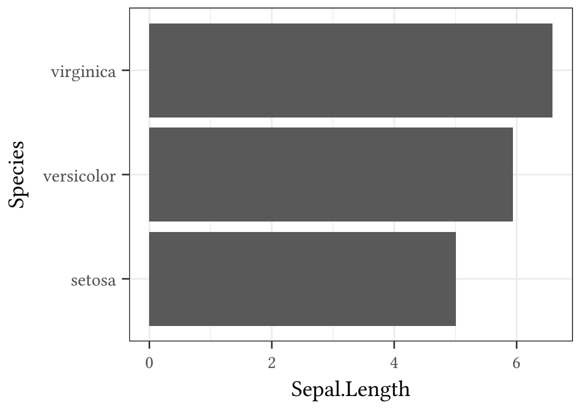
- The median: usually a thick line in the middle
- The "box", usually from the 25% to the 75% quantile
- The "whiskers", usually to the most extreme data points which are not more than 1.5× the interquartile range.

  If the data is normally distributed, then the interquartile range covers 1.35 standard deviations. Hence, 1.5× the interquartile range are 2.02 standard deviations. Outside the whiskers we should, hence, observe 4.3% (or about 5%) of all observations.

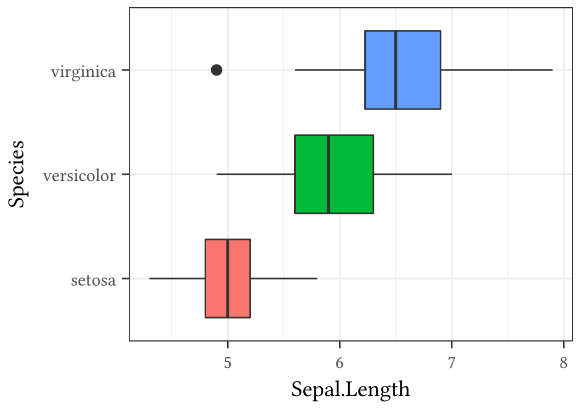
5.2.3 Barplot of means

On the right you see a barplot of means. I show this type only for completeness. Avoid, under all circumstances! You can show much more information in this space. On the right you see, for comparison, a boxplot.

```r
ggplot(iris,aes(x=Sepal.Length,y=Species)) +
    stat_summary(fun=mean,geom="bar")
```



```r
ggplot(iris, aes(Sepal.Length, y=Species, fill=Species)) +
    geom_boxplot() + theme(legend.position="none")
```

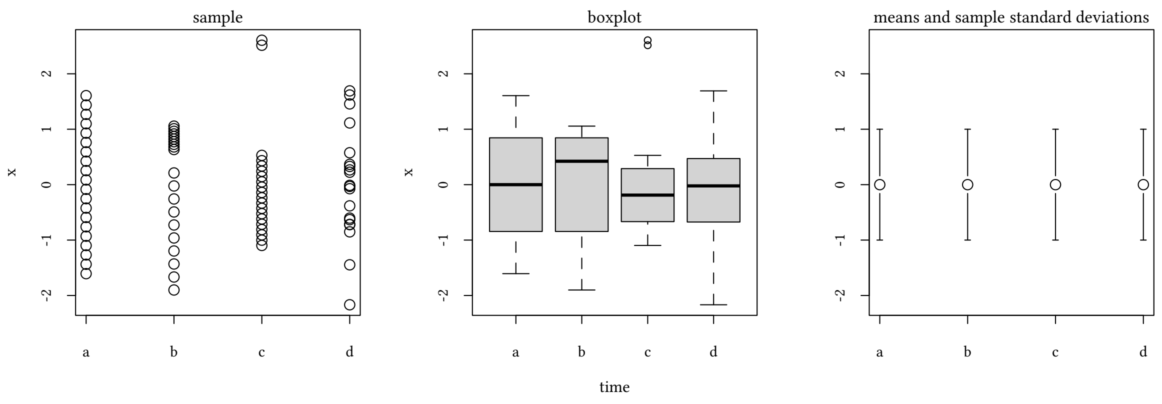


5.2.4 Means and standard deviation

Means and standard deviation are much less informative than boxplots. The following three graphs all show the same four distributions which all have identical sample means and standard deviations (right diagram). Still, scattergrams (left) and boxplots (middle) reveal that the four samples are quite different.

```r
set.seed(123)
xx<-as.data.frame(cbind(seq(0,2,length=20),c(seq(-8,2,length=10),seq(4,6,length=10)),
                        c(seq(0,1.8,length=18),4,4.1),rnorm(20)))
for(i in 1:4) {xx[,i]<-(xx[,i]-mean(xx[,i]))/sd(xx[,i])}
xx<-reshape(xx,direction="long",v.names="x",varying=list(1:4))
xx<-within(xx,{time<-as.factor(time);levels(time)<-letters[1:4]})
```

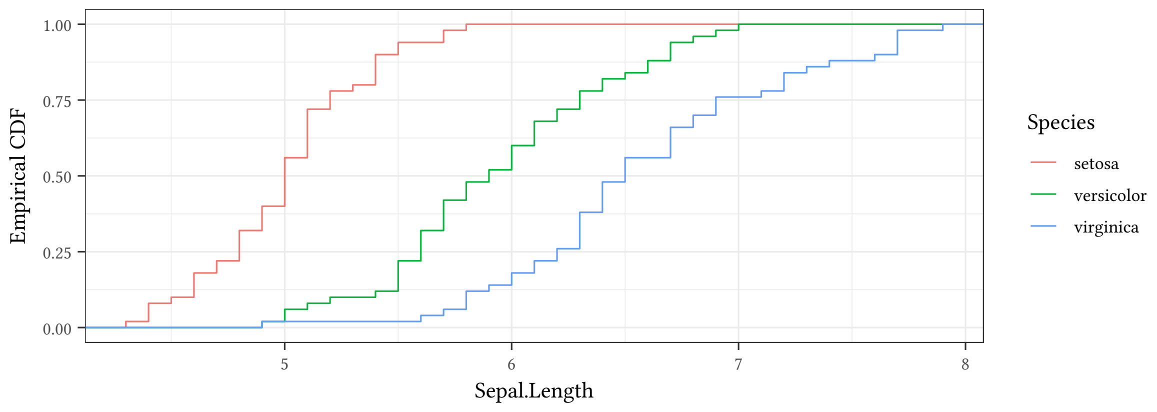
```r
ylim=range(c(xx$x))
par(mfrow=c(1,3))
with(xx,plot(x ~ as.integer(time),xaxt="n",ylim=ylim,xlab="",main="sample"))
axis(1,at=1:4,labels=letters[1:4])
boxplot(x ~ time,data=xx,ylim=ylim,main="boxplot")
library(plotrix)
dispData<-aggregate(x~time,
                    FUN=function(x) c(mean=mean(x),sd=sd(x)),
                    data=within(xx,time<-as.numeric(time)))
with(dispData,{
    plot(x[,"mean"] ~ time,dispData,xaxt="n",ylim=ylim,xlab="",ylab="",
         main="means and sample standard deviations")
    dispersion(1:4,x[,"mean"],x[,"sd"])
})
axis(1,at=1:4,labels=letters[1:4])
```



- Means and sample standard deviation may be misleading
- Boxplots provide more information, make fewer assumptions

5.2.5 Empirical cumulative distributions

```
ggplot(iris,aes(Sepal.Length,color=Species)) +
    stat_ecdf() +
    labs(y="Empirical CDF")
```



5.3 More on Dot-plots

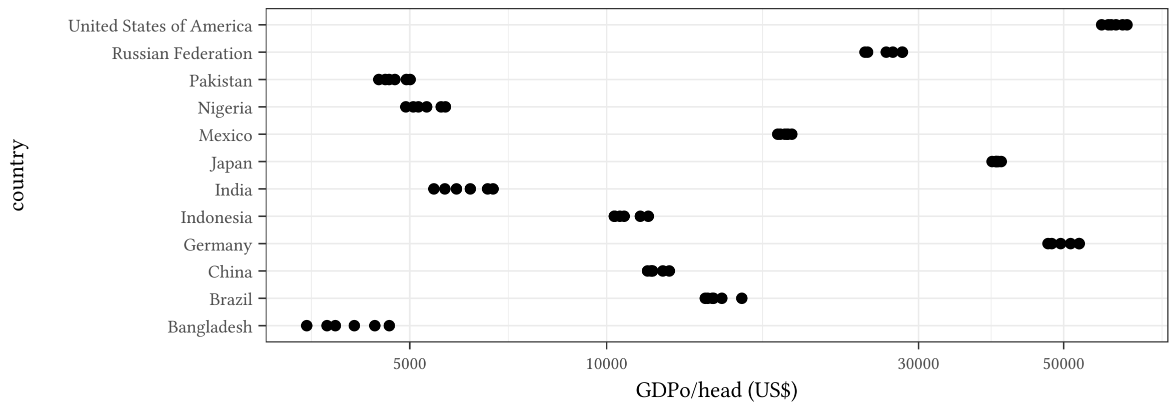
We use dot-plots if we have a small number of non-anonymous categories:

```
data(pwt10.0)
N <- 12
pwt10.0 %>%
    semi_join(pwt10.0 %>% ## find N most populous countries:
              group_by(country) %>%
              summarise(popM=median(pop,
                                    na.rm=TRUE)) %>%
              arrange(-popM) %>%
              top_n(N)) %>%
    filter(year>max(year)-6) %>%
    mutate(gdp = cgdpo/pop) %>%
    select(c("country","gdp","year")) -> pwt12
```

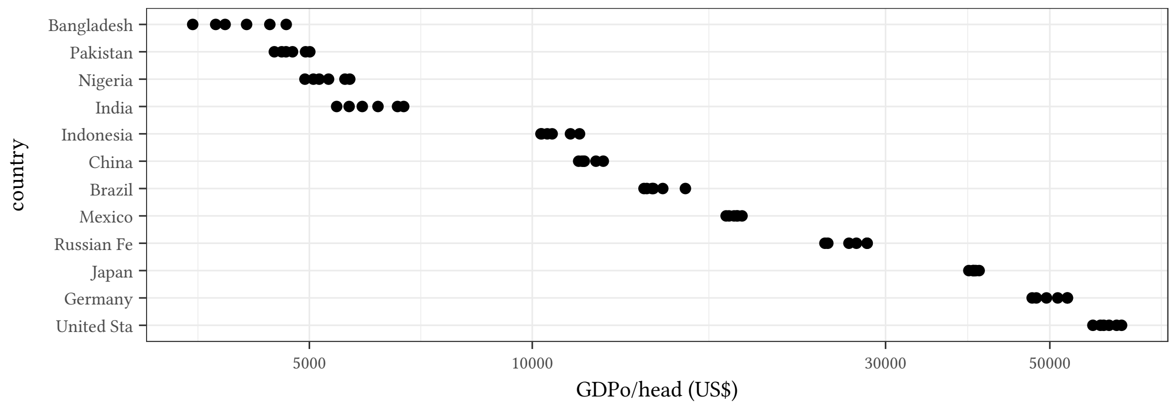
```
pwt12

            country       gdp year
BGD-2014 Bangladesh  3478.825 2014
BGD-2015 Bangladesh  3736.502 2015
BGD-2016 Bangladesh  3847.664 2016
BGD-2017 Bangladesh  4113.227 2017
BGD-2018 Bangladesh  4421.308 2018
BGD-2019 Bangladesh  4652.617 2019
BRA-2014     Brazil 16099.694 2014
BRA-2015     Brazil 15005.833 2015
BRA-2016     Brazil 14154.573 2016
BRA-2017     Brazil 14279.429 2017
```

```
BRA-2018     Brazil 14514.132 2018
BRA-2019     Brazil 14570.642 2019
CHN-2014      China 11733.252 2014
 [ reached 'max' / getOption("max.print") -- omitted 59 rows ]
```

```
ggplot(pwt12,aes(y=country,x=gdp)) + geom_point() +
    scale_x_log10()  + labs(x="GDPo/head (US\\$)")
```

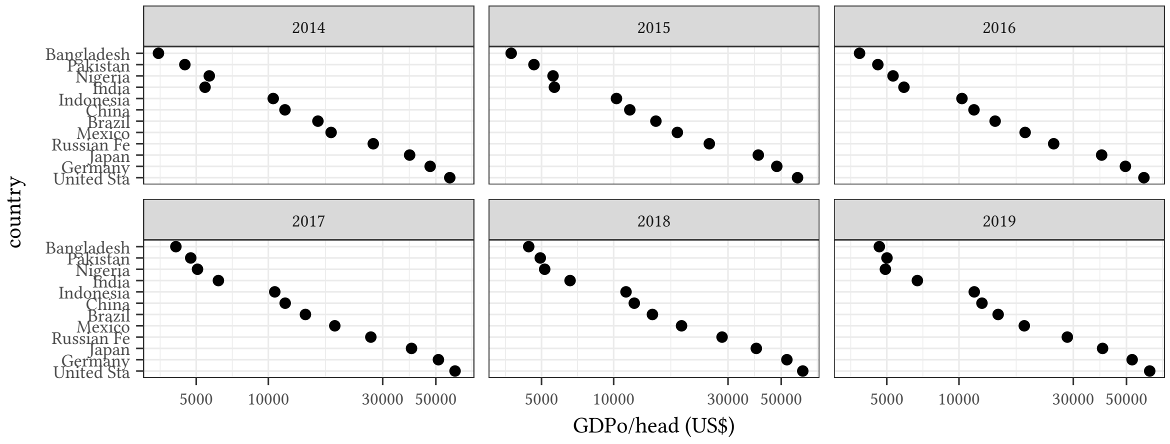


```
pwt12 %>%
    mutate(country = reorder(factor(substr(country,1,10)),-gdp)) %>%
    ggplot(aes(y=country,x=gdp)) + geom_point() +
    scale_x_log10()  + labs(x="GDPo/head (US\\$)")
```



Multiway dot-plots

```
pwt12 %>%
    mutate(country = reorder(factor(substr(country,1,10)),-gdp)) %>%
    ggplot(aes(y=country,x=gdp)) + geom_point() +
    scale_x_log10()  + labs(x="GDPo/head (US\\$)") +
    facet_wrap(vars(year))
```



5.4 Summary

Histograms

+ Everybody understands them
− Don't reveal small differences
− Depend on breaks

Densities

+ Easy to understand
− Need assumptions (must be estimated)
− Depend on bandwidth

Conditional density plots

+ Easy to understand
+ Reveals even small differences between distributions
− Needs assumptions (must be estimated)

Boxplot

+ Shows summary statistics
− Aggregates data

Barplot of means

− Uses a lot of space to show a small amount of information.

ECDF

+ Provides a lot of information
+ Doesn't depend much on parameters

+ Reveals even small differences between distributions
– Not so easy to understand

Q-Q Plot

+ Provides a lot of information
+ Doesn't depend much on parameters
+ Reveals even small differences between distributions
– Only compares two variables

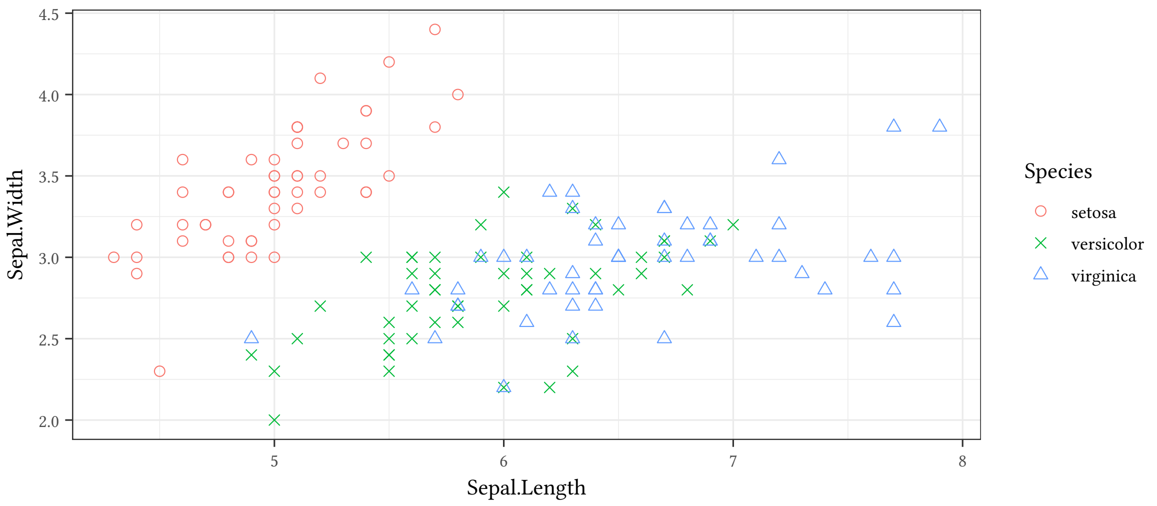
Dot-Plot

+ Provides detailed information
+ Doesn't depend much on parameters
– Requires a small number of observations

5.5 Two continuous variables

5.5.1 Scatterplot

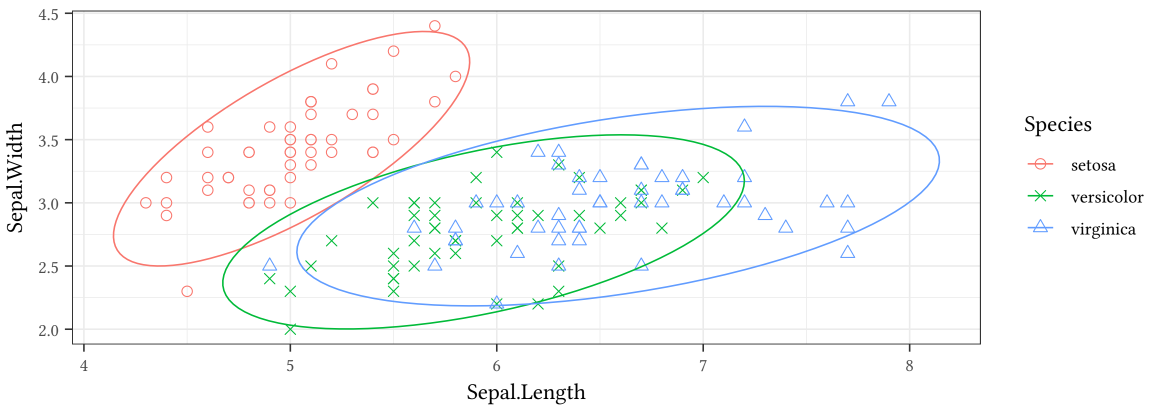
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,color=Species,shape=Species)) +
    geom_point()
```



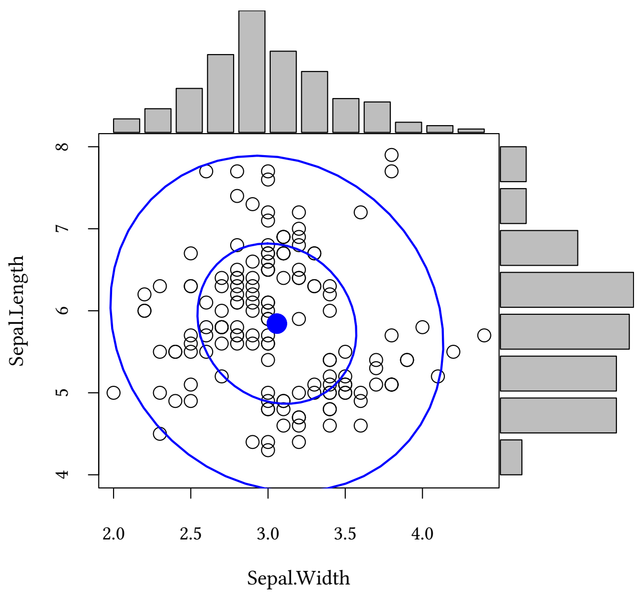
With larger data frames scatterplots might provide too much information:

5.5.2 Scatterplot with data ellipses

```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,color=Species,shape=Species)) +
    geom_point() + ggpubr::stat_conf_ellipse(bary=FALSE)
```



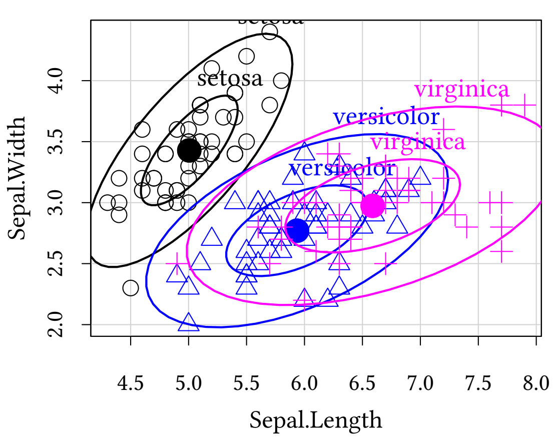
```
library(car)
attach(iris)
xhist <- hist(Sepal.Width, breaks=10,plot=FALSE)
yhist <- hist(Sepal.Length, breaks=10,plot=FALSE)
xrange <- range(xhist$breaks)
yrange <- range(yhist$breaks)
layout(rbind(c(2,0),c(1,3)),
       widths=c(4,1), heights=c(1,4))
par(mar=c(4,4,0,0))
plot(Sepal.Width, Sepal.Length,
     xlim=xrange, ylim=yrange)
dataEllipse(Sepal.Width,Sepal.Length,
            levels=c(.5,.95),plot.points=FALSE)
par(mar=c(0,4,0,0))
barplot(xhist$counts, axes=FALSE)
par(mar=c(4,0,0,0))
barplot(yhist$counts, axes=FALSE,
        horiz=TRUE)
detach(iris)
```



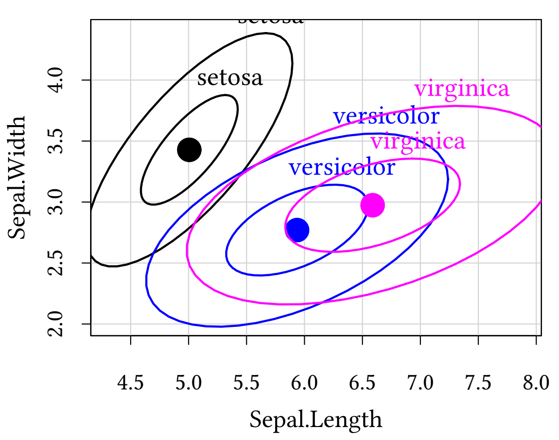
We use `hist` to calculate the range of the plot and to prepare the barplot:

```
attach(iris)
xhist <- hist(Sepal.Width, breaks=10,plot=FALSE)
yhist <- hist(Sepal.Length, breaks=10,plot=FALSE)
xhist$breaks

 [1] 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4

xhist$counts

 [1]  4  7 13 23 36 24 18 10  9  3  2  1

detach(iris)
```

```
library(car)
attach(iris)
dataEllipse(Sepal.Length,Sepal.Width,
            groups=Species,levels=c(.5,.95))
detach(iris)
```



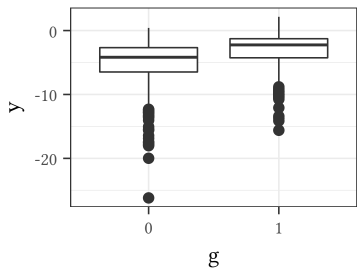
```
library(car)
attach(iris)
dataEllipse(Sepal.Length,Sepal.Width,
            groups=Species,levels=c(.5,.95),
            draw=TRUE,plot.points=FALSE,add=FALSE)
detach(iris)
```



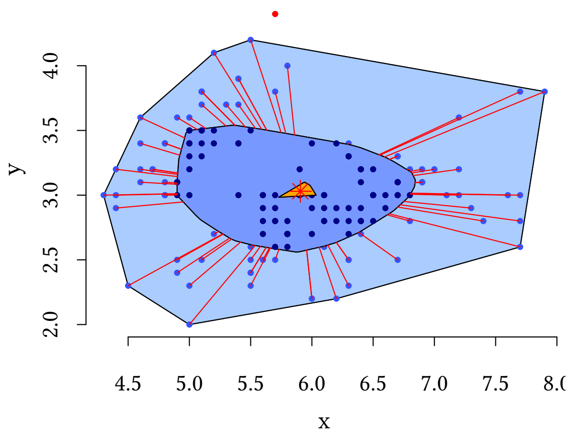
5.5.3 Bagplot

P. J. Rousseeuw, I. Ruts, J. W. Tukey (1999)

- The dark-blue area: The “bag”. This area contains 50% of all observations.
- The light-blue area: Contains all points which are in the bag 3 times expanded.
- Points outside the light-blue area are considered outliers.

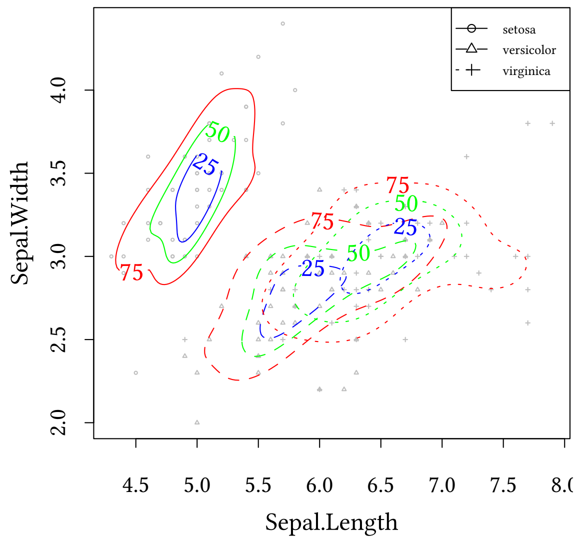


```
library(aplpack)
with(iris,bagplot(Sepal.Length,Sepal.Width))
```

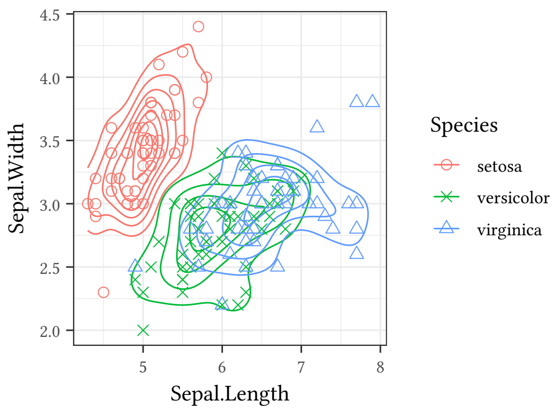


5.5.4 Kernel densities

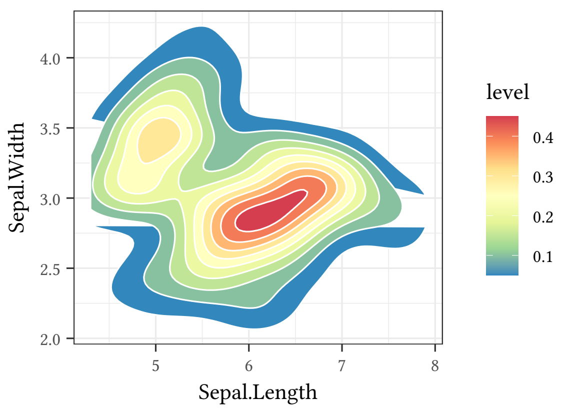
```
library(ks)
iris %>%
    select(Sepal.Length,Sepal.Width,Species) ->
    data2
dlply(data2,.(Species),
      function(d) {
          kde(d[,1:2])
      }) -> kdeList
with(data2,
     plot(Sepal.Length,Sepal.Width,cex=.2,
          col="gray",pch=as.numeric(Species)))
for(i in 1:length(kdeList))
    plot(kdeList[[i]],add=TRUE,lty=i,
         col.fun=function(n){rainbow(n)})
legend("topright",
       lty=1:length(kdeList),
       pch=1:length(kdeList),
       names(kdeList),cex=.5)
```



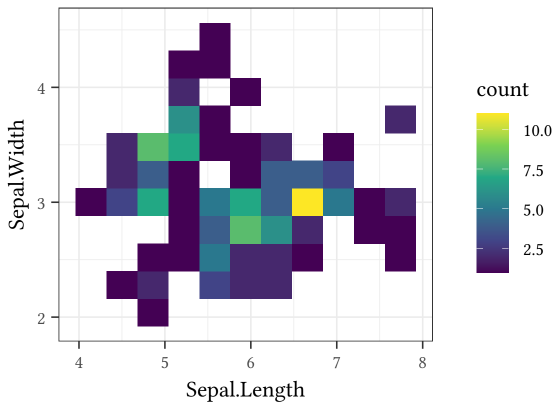
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,
                color=Species,shape=Species)) +
    geom_density_2d() +
    geom_point()
```



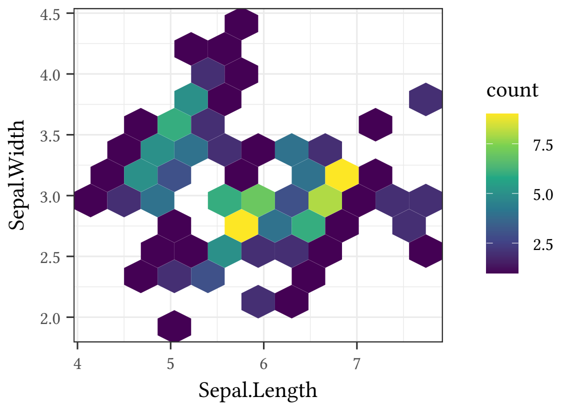
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    stat_density_2d(aes(fill = ..level..),
                    geom = "polygon",
                    colour="white") +
    scale_fill_distiller(palette= "Spectral")
```



```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_bin2d(bins=10) +
    scale_fill_continuous(type = "viridis")
```



```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width)) +
    geom_hex(bins=10) +
    scale_fill_continuous(type = "viridis")
```

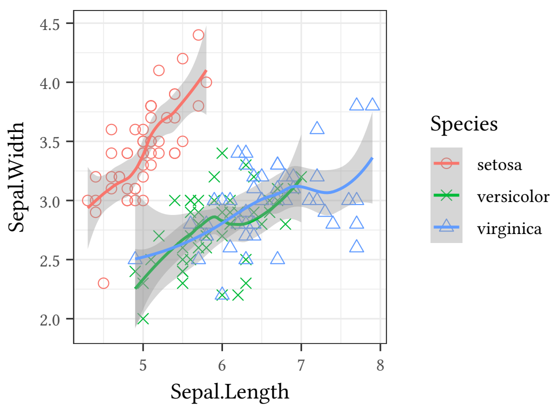


6 Continuous data, causal relations, other problems

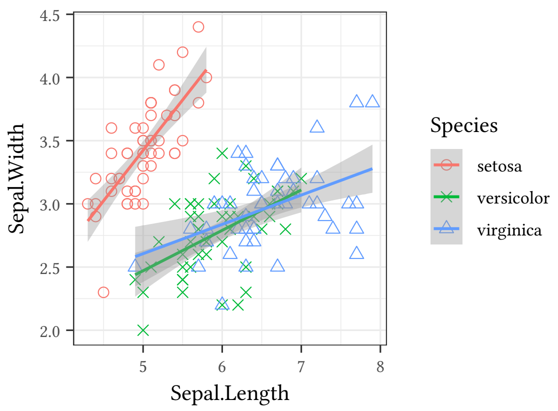
6.1 Causal relations

6.1.1 Smooth lines

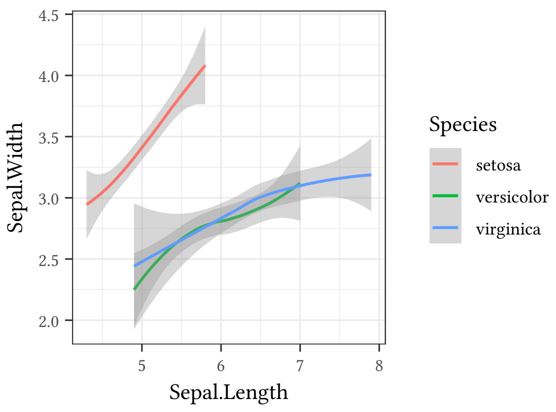
```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,
                color=Species,shape=Species)) +
    geom_point() + geom_smooth()
```



```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,
                color=Species,shape=Species)) +
    geom_point() + geom_smooth(method="lm")
```

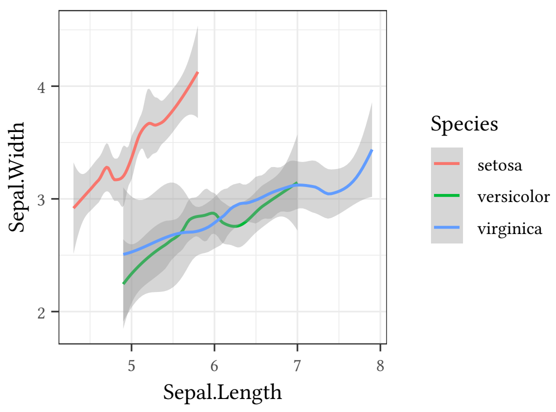


```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,
                color=Species,shape=Species)) +
    geom_smooth(span=1)
```



How smooth?

```
ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,
                color=Species,shape=Species)) +
    geom_smooth(span=.5)
```



6.1.2 GAM

Loess (locally estimated scatterplot smoothing) only relates one variable to a smooth function of one other variables. What if there are more variables?

For more complex relationships (and as an extension of the linear model) we can use GAM (generalised additive models).

Linear Regression:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + u$$

GAM (Generalised additive model):

$$Y = \beta_0 + s_1(X_1) + s_2(X_2) + \ldots + \beta_k X_k \ldots + u$$

```
est.ols <- lm(testscr ~ elpct + avginc + str,data=Caschool)
library(mgcv)
est.gam <- gam(testscr ~ s(elpct) + s(avginc) + str,data=Caschool)
```

Here is the output for the standard OLS model:

```
summary(est.ols)


Call:
lm(formula = testscr ~ elpct + avginc + str, data = Caschool)

Residuals:
    Min      1Q  Median      3Q     Max 
-42.800  -6.862   0.275   6.586  31.199 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 640.31550    5.77489 110.879   <2e-16 ***
elpct        -0.48827    0.02928 -16.674   <2e-16 ***
avginc        1.49452    0.07483  19.971   <2e-16 ***
str          -0.06878    0.27691  -0.248    0.804    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.35 on 416 degrees of freedom
Multiple R-squared:  0.7072,Adjusted R-squared:  0.7051 
F-statistic: 334.9 on 3 and 416 DF,  p-value: < 2.2e-16
```

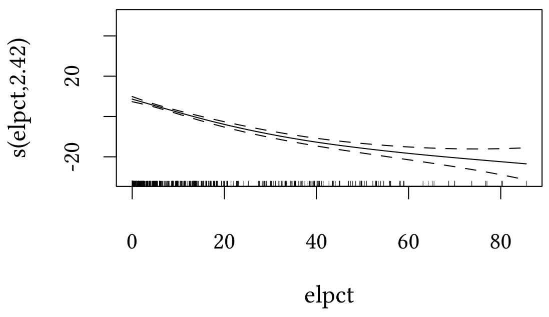
The output for a GAM is similar to the output for OLS. Of course, the splines (here for `elpct` and `avginc`) are not shown. The output provides only the result of an F-test and the estimated degrees of freedom (edf).

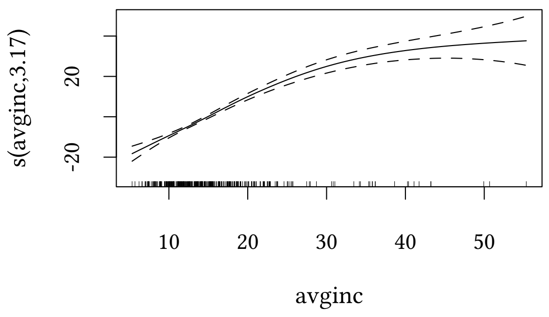
```
summary(est.gam)


Family: gaussian 
Link function: identity 

Formula:
testscr ~ s(elpct) + s(avginc) + str

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 656.9103     5.3040 123.852   <2e-16 ***
str          -0.1402     0.2689  -0.521    0.602    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
            edf Ref.df      F p-value    
s(elpct)  2.416  3.023  87.53  <2e-16 ***
s(avginc) 3.171  3.983 116.57  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
R-sq.(adj) =  0.731   Deviance explained = 73.5%
GCV = 99.459  Scale est. = 97.662    n = 420
```

Here are the functions s_1 and s_2:

```
plot(est.gam,pages=0)
```





GAM permits interactions of splines:

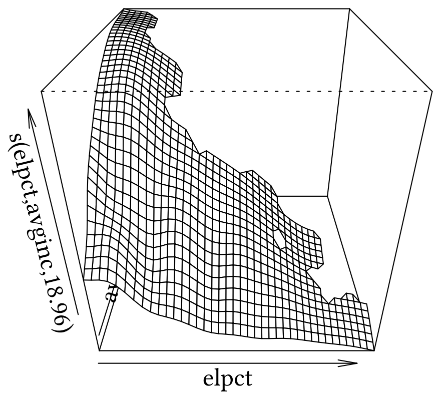
```
library(mgcv)
est2.gam <- gam(testscr ~ s(elpct,avginc) + str,data=Caschool)
```

```
summary(est2.gam)


Family: gaussian
Link function: identity

Formula:
testscr ~ s(elpct, avginc) + str

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 658.0740     5.3777 122.371   <2e-16 ***
str          -0.1995     0.2728  -0.731    0.465
---
```

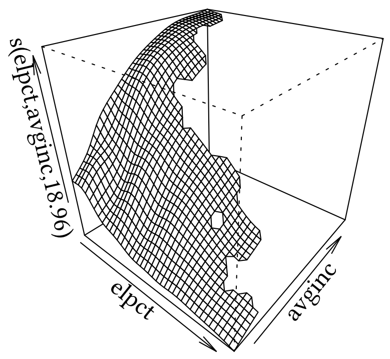
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                  edf Ref.df     F p-value    
s(elpct,avginc) 18.96  23.79 47.72  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.743   Deviance explained = 75.6%
GCV = 98.084  Scale est. = 93.19     n = 420
```

Now the spline is a smooth surface:

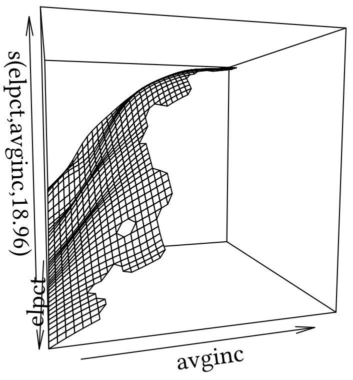
```
plot(est2.gam,pages=1,pers=TRUE,theta=0)
```



```
plot(est2.gam,pages=1,pers=TRUE,theta=40)
```

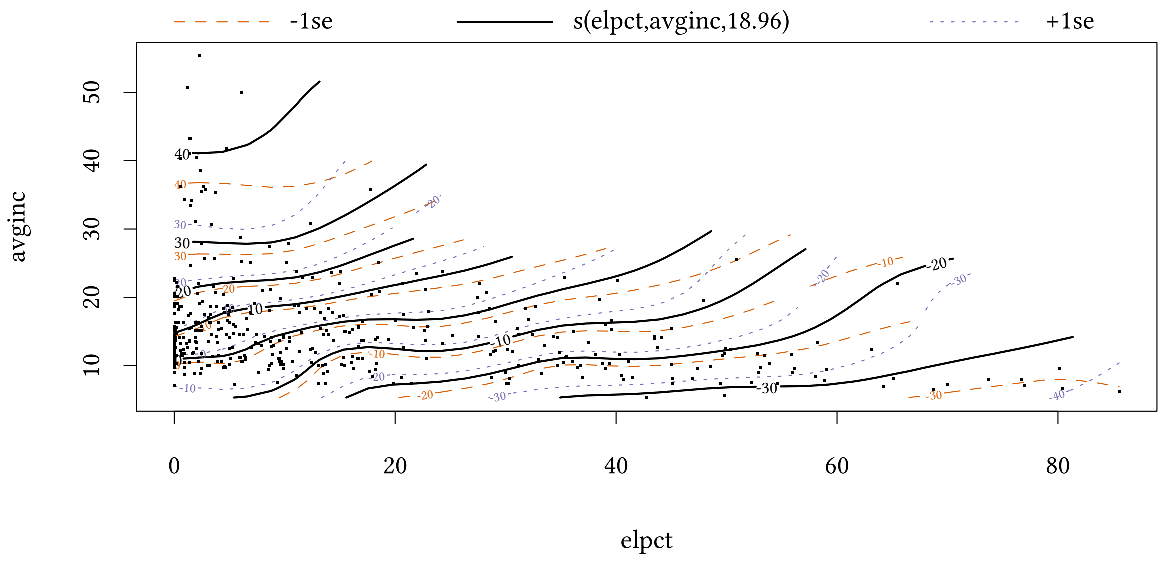


```
plot(est2.gam,pages=1,pers=TRUE,theta=75,phi=5)
```



The surface can also be shown as contours:

```
plot(est2.gam,pages=1)
```

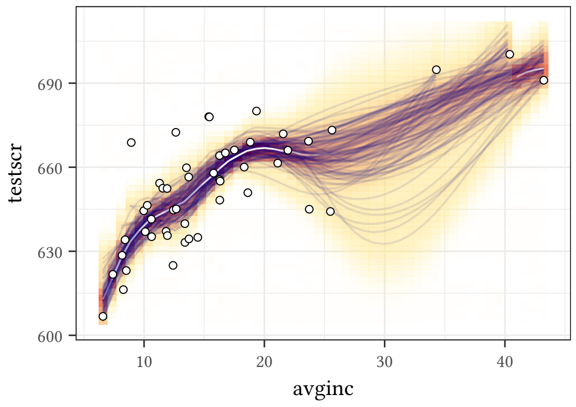


6.1.3 Visually weighted Regression

```
source("~/R/vwreg.R")
set.seed(123)
CaPart<-Caschool[sample(1:nrow(Caschool),50),]
```

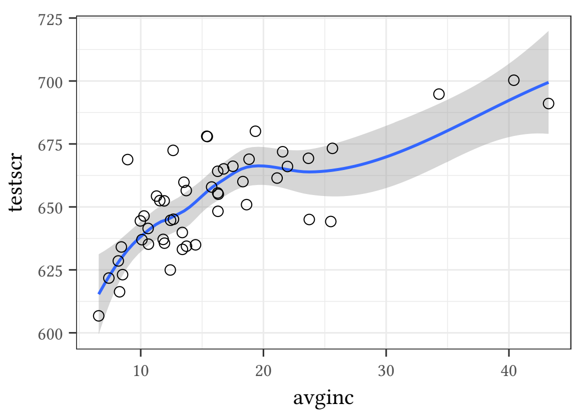
- Solomon Hsiang (2012). Visually weighted Regression.
- Felix Schonbrodt (2012): Implementation in R.

```
vwReg(testscr~avginc,data=CaPart,B=100,
      spag=TRUE,slices=50)
```



Same situation with loess+standard deviation:

```
ggplot(CaPart,aes(x=avginc,y=testscr)) +
    geom_smooth() +
    geom_point(shape=1)
```



6.1.4 Summary: Two continuous variables

Scatterplot

+ makes no assumptions
– with a large dataset the graph might be cluttered
– with categorical data points might superimpose

Data ellipses

– assumes a linear relationship

Bagplot

– not very well known

Kernel densities

+ easy to understand
– relies on assumptions (must be estimated, depend on bandwidth)

Regression line

– Assumes a linear causal relationship

Loess/GAM/VWReg

– Assume causal (not necessarily linear) relationship

6.2 Other problems

6.2.1 Paired data

Sometimes two-dimensional data comes in pairs where both elements can be compared with each other. One value might be recorded before, the other after a treatment.

If the data are highly correlated then a standard scatterplot (left diagram) wastes a lot of space top left and bottom right from the 45°-line.

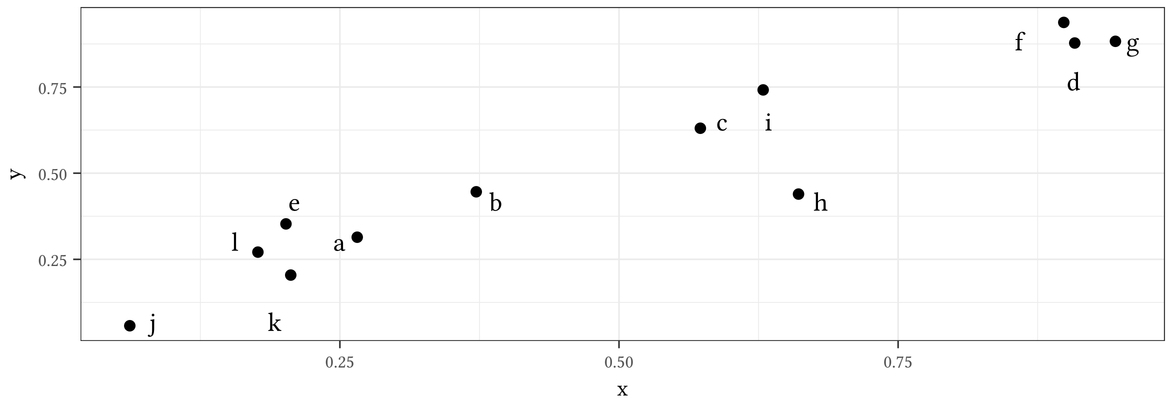
The Tukey mean-difference plot (also known as Bland-Altman plot) basically rotates the diagram by 45° and, thus, can save space. (This plot aims at showing agreement of the two elements of the pairs and, hence, also shows the mean of the differences ± two standard deviations.)

The bumpchart presents essentially the same information, but with a focus on the identity of the observations. We would usually not do this for anonymous observations, but, e.g. if observations are for countries or for cities.

Create some paired data:

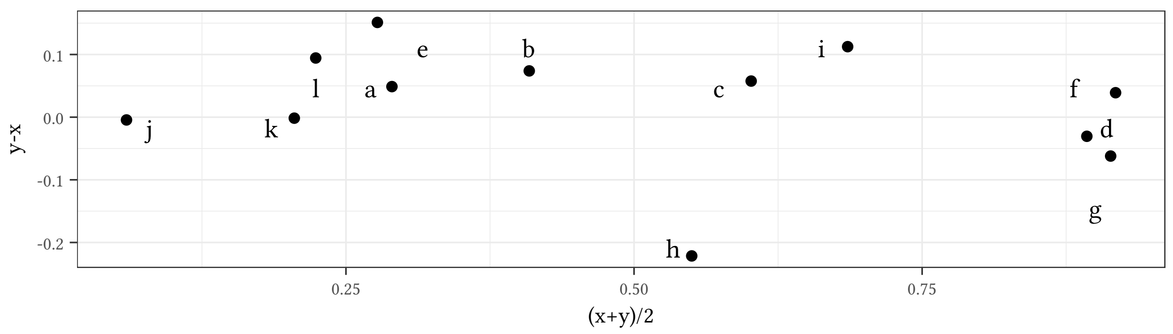
```
set.seed(1)
N<-12
x<-runif(N)
y<-x+.1*rnorm(N)
pairD <- data.frame(x=x,y=y,g=letters[1:N])
```

```
pairD %>% ggplot(aes(x=x,y=y)) + geom_point() +
    directlabels::geom_dl(aes(label=g),
                          method="smart.grid")
```



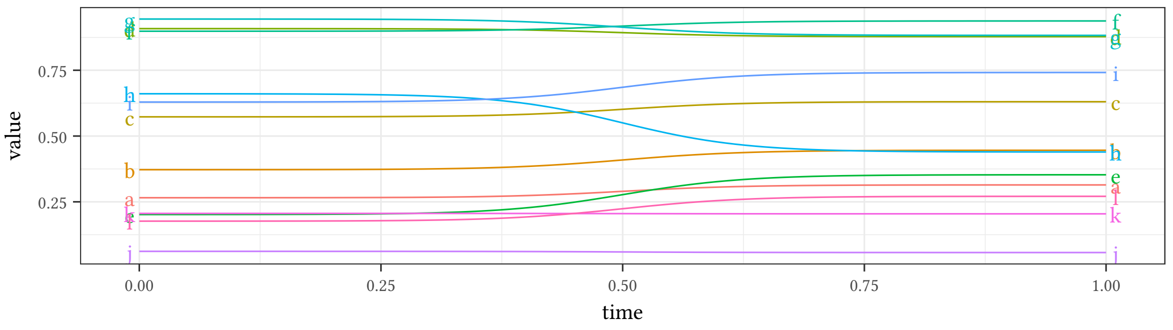
Bland-Altman / Tukey mean-difference:

```
pairD %>%
    mutate(xyMean = (x+y)/2, yxDiff = (y-x)) %>%
    ggplot(aes(x=xyMean,y=yxDiff)) + geom_point() +
    labs(x="(x+y)/2",y="y-x") +
    directlabels::geom_dl(aes(label=g),
                          method="smart.grid")
```



Bump-plot:

```
pairD %>% tidyr::pivot_longer(cols=c("x","y")) %>%
   mutate(time=ifelse(name=="x",0,1)) %>%
   ggplot(aes(x=time,y=value,color=g)) +
   ggbump::geom_bump() +
   geom_text(data=pairD,aes(x=-.01,y=x,label=g)) +
   geom_text(data=pairD,aes(x=1.01,y=y,label=g)) +
   theme(legend.position="none")
```



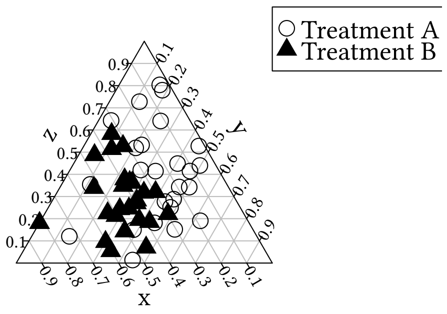
6.2.2 Three-dimensional simplex

Three dimensional variables are notoriously difficult to present. However, quantities like prices and probabilities can often be conveniently represented in a simplex:

```
set.seed(123)
data3 <-matrix(runif(150),ncol=3)
type<-data3[,1]>.5    # <- add groups
data3<-data3/rowSums(data3) # <- normalise
colnames(data3)<-c("x","y","z")
data3
```

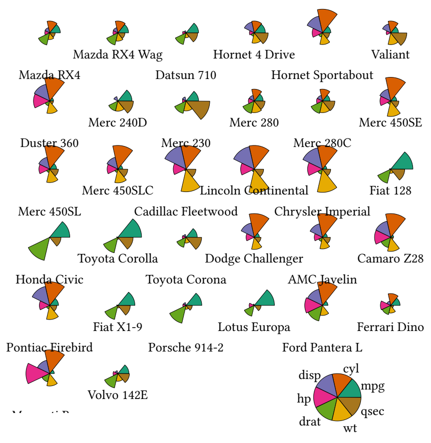
```
               x            y          z
 [1,] 0.30809754 0.0491014380 0.64280102
 [2,] 0.50424783 0.2828580196 0.21289415
 [3,] 0.24106888 0.4709212352 0.28800989
 [4,] 0.45065923 0.0622128465 0.48712793
 [5,] 0.47394996 0.2826906163 0.24335942
 [6,] 0.03987656 0.1807812499 0.77934219
 [7,] 0.33635678 0.0812264534 0.58241676
 [8,] 0.39584570 0.3341408730 0.27001343
 [9,] 0.29692218 0.4819404184 0.22113740
[10,] 0.46680404 0.3828188688 0.15037709
[11,] 0.37416520 0.2600901850 0.36574461
[12,] 0.53370870 0.1116555756 0.35463572
[13,] 0.60375503 0.3421393873 0.05410558
 [ reached getOption("max.print") -- omitted 37 rows ]
```

```
triax.plot(data3,show.grid=TRUE,pch=16*type+1,
           cex.ticks=.7)
legend(1,1,c("Treatment A","Treatment B"),
       pch=c(1,17))
```



6.2.3 Stars

```
stars(mtcars[, 1:7], len = 0.8, key.loc = c(12, 1.5),draw.segments = TRUE,cex=.5)
```



7 Lattice

7.1 Multiway xyplots

Sometimes we want to display one type of diagram separately for different levels of a factor. Here is an example:

Example: development of investment share (`ci`) over time (`year`), separately for each `country`.

Get a subset of the data (six largest countries, later than 2001) from the Penn World Table:

```
library(pwt)
lattice.options(default.args=list(as.table=TRUE))
data(pwt6.3)
```

Add average population to data:

```
pwt6.3 |>
    group_by(country) |>
    summarise(popM=mean(pop)) |>
    filter(country!="China Version 2") |>
    arrange(-popM) |>
    head(6) -> largeCountries
##
pwt6.3 |>
    semi_join(largeCountries) |>
    filter(year>2001) -> pwSub
```

Give two countries a shorter name:

```
levels(pwSub$country)[grep("United States",levels(pwSub$country))]<-"U.S.A."
levels(pwSub$country)[grep("China",levels(pwSub$country))]<-"China"
```

reorder the countries. The order of the factor is used later in the plots. Here we order according to the median of ci:

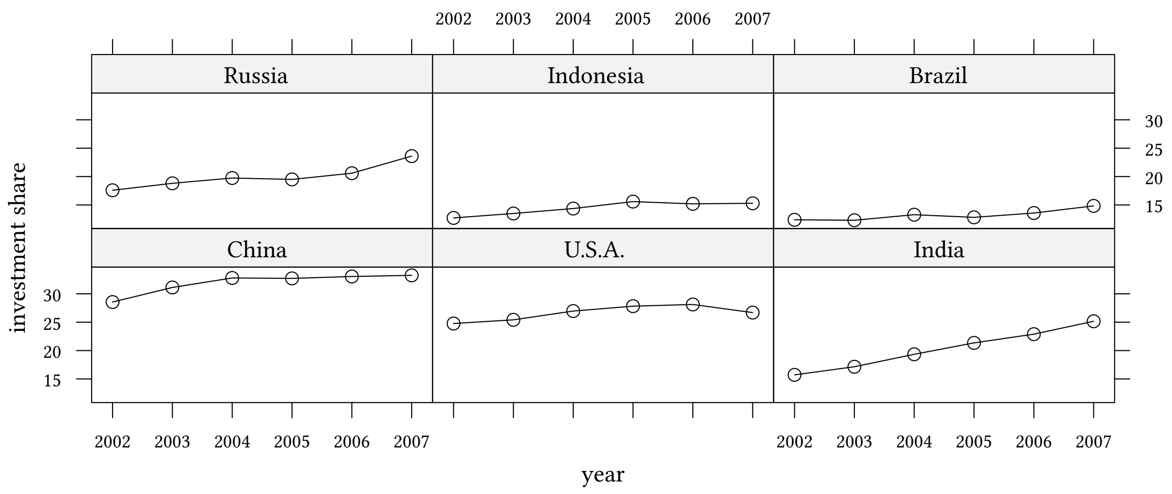
```
pwSub<-within(pwSub,country<-reorder(factor(country),pwSub$ci,
                                     function(x) -median(x,na.rm=TRUE)))
```

Sorting the data along year and country makes it easier to draw connect lines lateron:

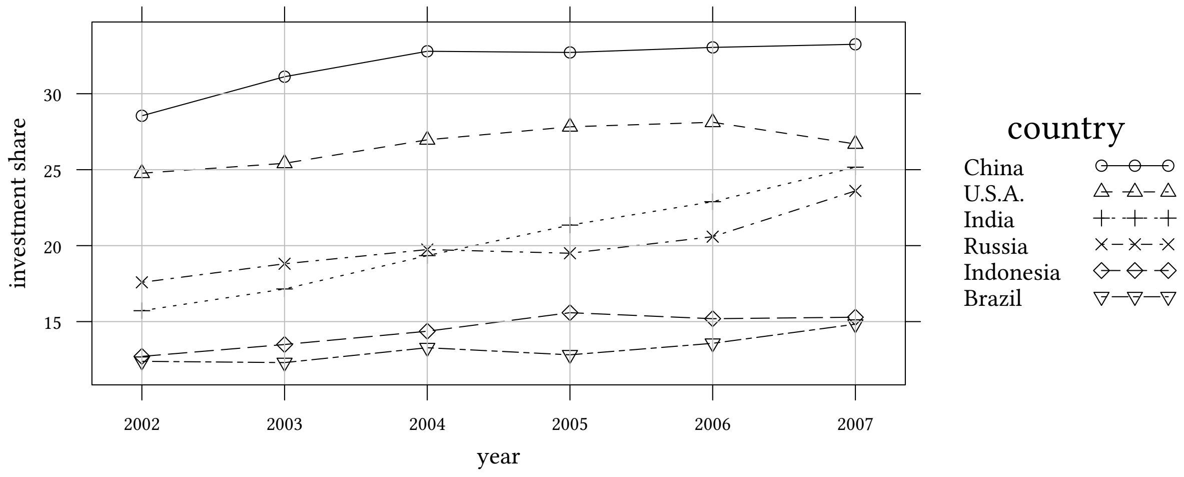
```
pwSub.pw<-pwSub[with(pwSub,order(country,year)),]
```

```
pwSub.pw[,c("country","year","pop","ci")]

         country year       pop       ci
CHN-2002   China 2002 1284275.9 28.54705
CHN-2003   China 2003 1291496.0 31.11946
CHN-2004   China 2004 1298847.6 32.79388
CHN-2005   China 2005 1306313.8 32.71553
CHN-2006   China 2006 1313973.7 33.04857
CHN-2007   China 2007 1321851.9 33.25413
USA-2002  U.S.A. 2002  287501.5 24.76686
USA-2003  U.S.A. 2003  289985.8 25.41723
USA-2004  U.S.A. 2004  292805.6 26.96360
USA-2005  U.S.A. 2005  295583.4 27.82229
 [ reached 'max' / getOption("max.print") -- omitted 26 rows ]
```

```
lattice::xyplot(ci ~ year| country,data=pwSub.pw,
       ylab="investment share",type="b")
```

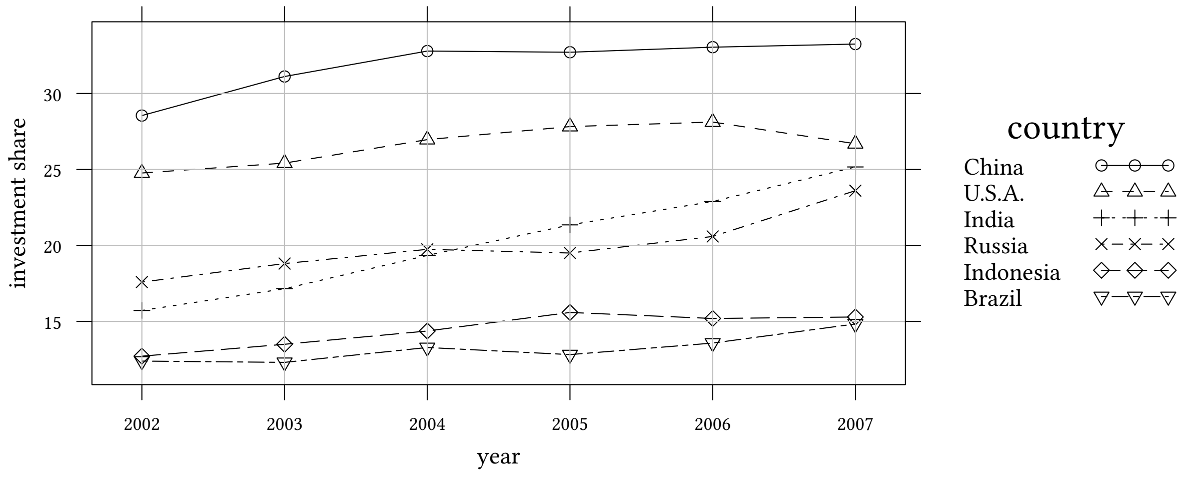


```
xyplot(ci ~ year,group=country,data=pwSub.pw,ylab="investment share",type="b",
             auto.key=list(space="right",title="country"))
```



The legend of the previous plot does not show the lines.

```
xyplot(ci ~ year,group=country,data=pwSub.pw,ylab="investment share",type="b",
              auto.key=list(space="right",title="country",lines=TRUE))
```



7.2 Syntax

The data we want to display in our lattice is described with the help of a formula:

Graphs with variables on the vertical and horizontal axis:

- $\boxed{\text{vertical} \sim \text{horizontal}}$ creates only one graph
- $\boxed{\text{vertical} \sim \text{horizonal} \mid \text{conditioning variable}}$ creates for each level of the conditioning variable one panel with one graph.

- `vertical ~ horizontal , group=grouping variable` creates only one panel and superimposes within this panel graphs for each level of the grouping variable.
- `vertical ~ horizontal | conditioning variable , group=grouping variable` creates for each level of the conditioning variable one panel. Within these panels graphs for each level of the grouping variable are superimposed.

Graphs with variables only on the horizontal axis (examples would be density plots, histograms, etc.):

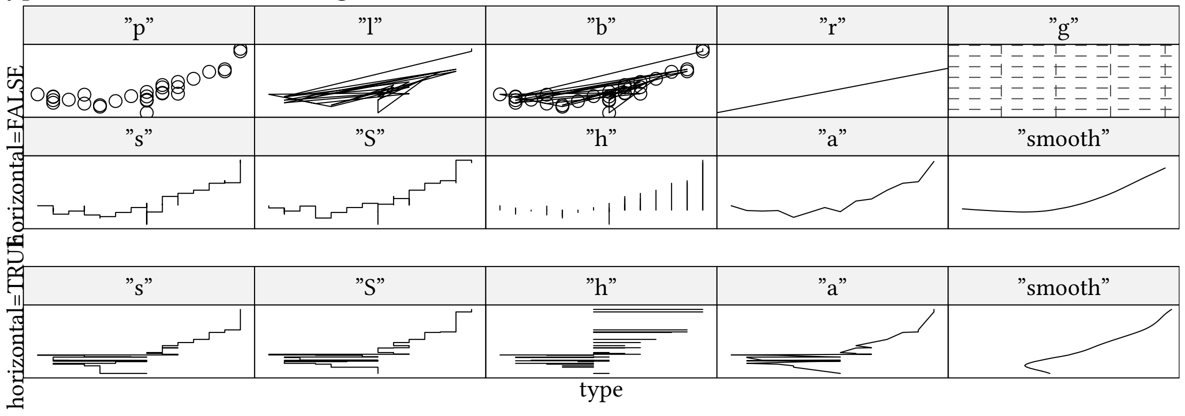
…when R creates values for vertical axis (e.g. histogram, densityplot, ecdfplot):

- `~ horizontal` creates only one graph
- `~ horizonal | conditioning variable` creates for each level of the conditioning variable one panel with one graph.
- `~ horizontal , group=grouping variable` creates only one panel and superimposes within this panel graphs for each level of the grouping variable.
- `~ horizontal | conditioning variable , group=grouping variable` creates for each level of the conditioning variable one panel. Within these panels graphs for each level of the grouping variable are superimposed.

Several variables on the horizontal axis

- `...~ h1 + h2 ...` shows two variables `h1` and `h2`

Types of lines The parameter `types` determines `how` points are displayed: `type="b"` or `type=c("b","smooth","g")`

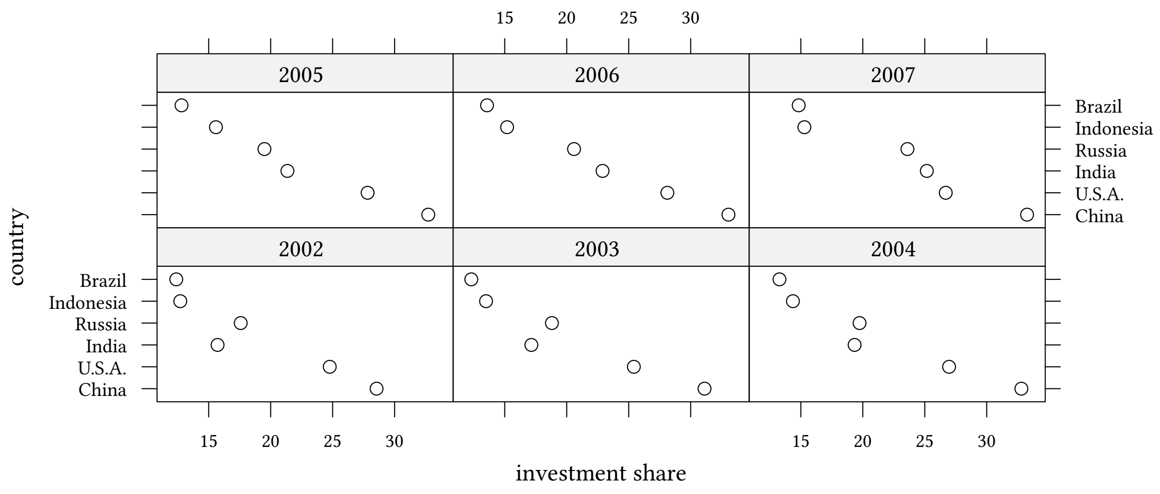


7.3 Multiway continued

Instead of having different panels for different countries, we could also have different panels for different years:

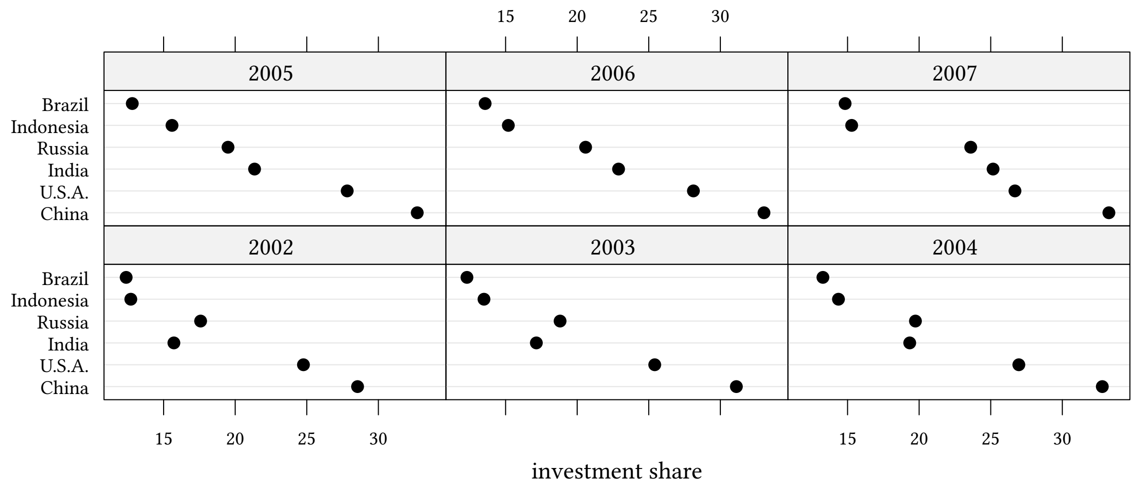
In the next plot we swap variables. We apply `factor` to year, so that it appears as a text in the shingles.

```
lattice::xyplot(country ~ ci| factor(year),data=pwSub.pw,xlab="investment share")
```



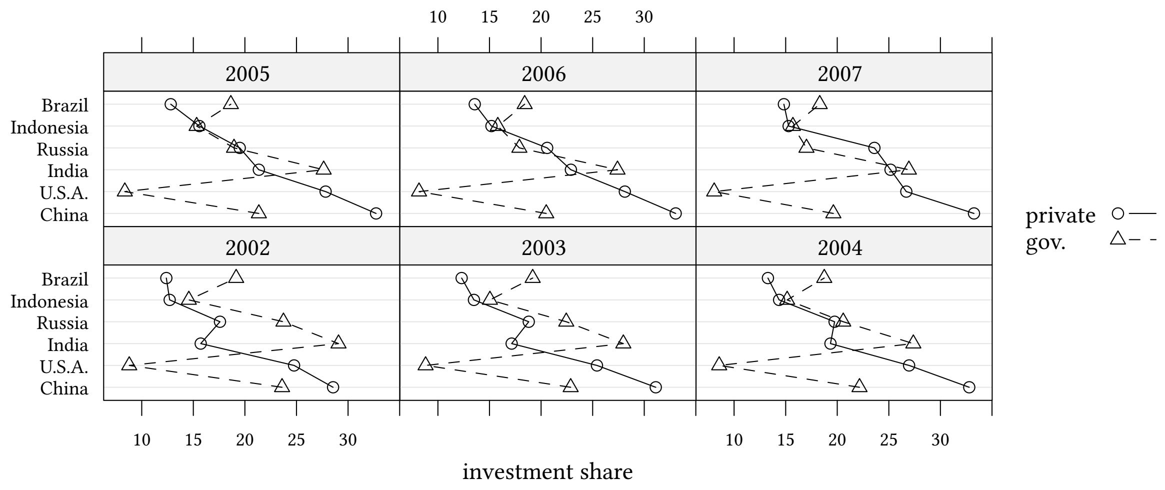
If the vertical variable (`country` in this case) is a factor, then `dotplot` generates even nicer graphs:

```
lattice::dotplot(country ~ ci| factor(year),data=pwSub.pw,
        xlab="investment share",horizonal=TRUE)
```



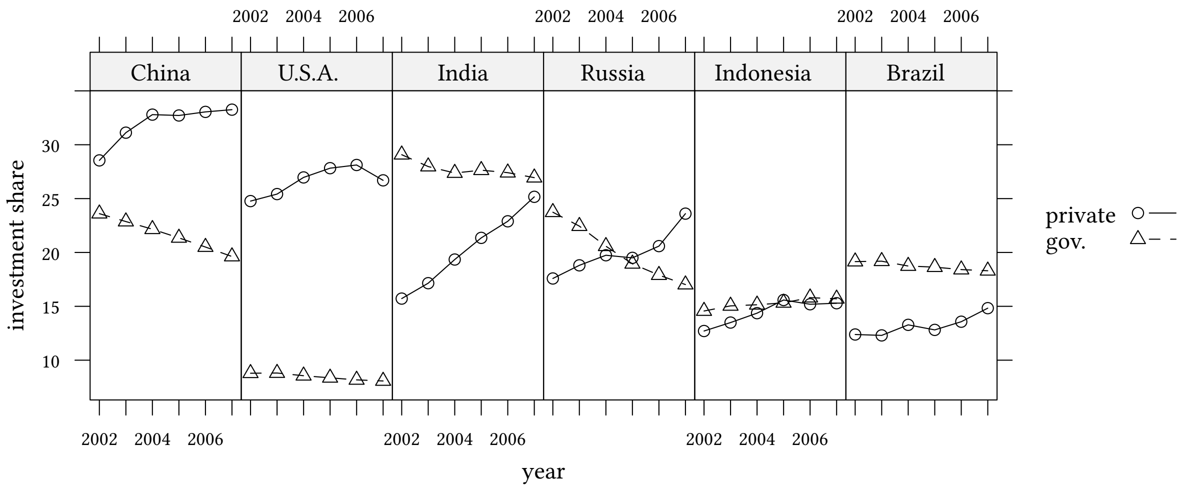
We can, of course, show more than one variable on the horizontal axis:

```
keys<-list(text=c("private","gov."),space="right",lines=TRUE,size=2,between=.5)
lattice::dotplot(country ~ ci+cg| factor(year),data=pwSub.pw,xlab="investment share",
        horizonal=TRUE,auto.key=keys,t="b")
```



Certainly, we can also have more than one variable on the vertical axis:

```
lattice::xyplot(ci+cg ~ year| country,layout=c(6,1),data=pwSub.pw,
       ylab="investment share",auto.key=keys,t="b")
```

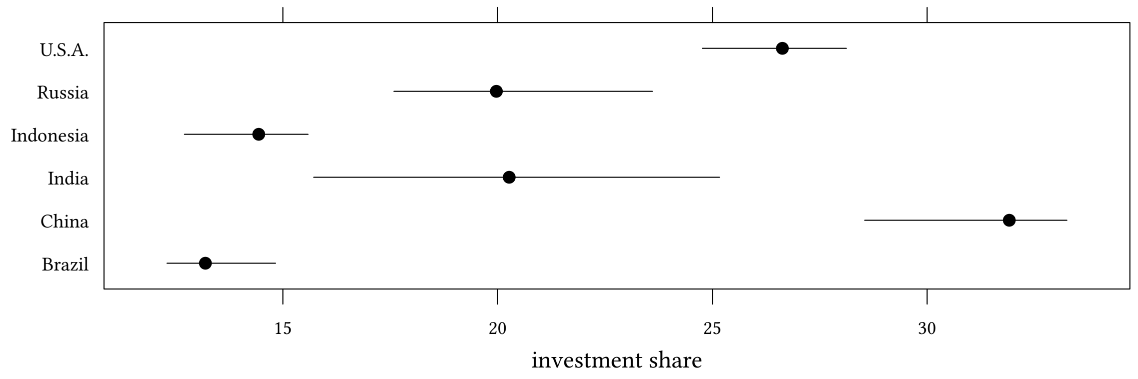


Segment plots Sometimes we have to plot segments. Here we plot a range of the minimum investment share to the maximum investment share.

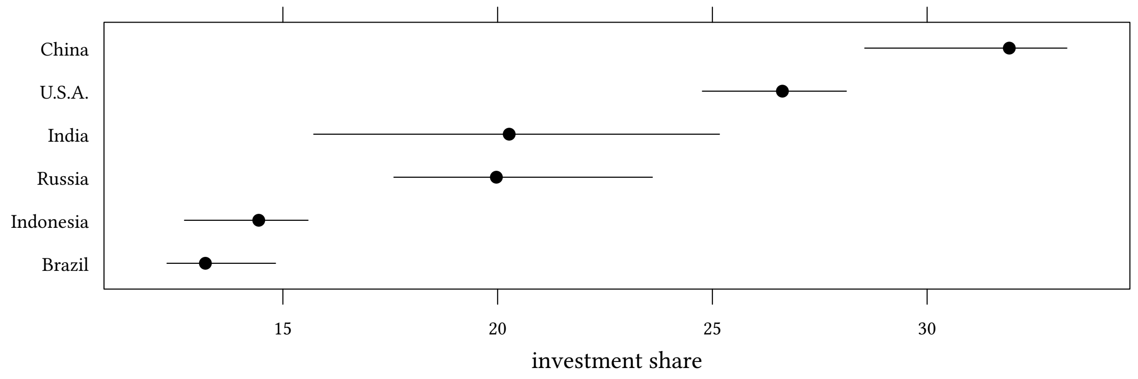
```
library(latticeExtra)
pwSub2<-as.data.frame(t(sapply(by(pwSub.pw,list(pwSub.pw$country),function(x)
    c(min=min(x$ci),mean=mean(x$ci),max=max(x$ci))),c)))
pwSub2
```

```
               min     mean      max
China     28.54705 31.91310 33.25413
U.S.A.    24.76686 26.63036 28.11900
India     15.71790 20.26952 25.16494
Russia    17.58459 19.97048 23.60284
Indonesia 12.70857 14.43829 15.58585
Brazil    12.30157 13.19624 14.82930
```

```
pwSub2<-within(pwSub2,{country<-factor(rownames(pwSub2))})
latticeExtra::segplot(country ~ min+max,centers=mean,draw.bands=FALSE,xlab="investment share",d
```



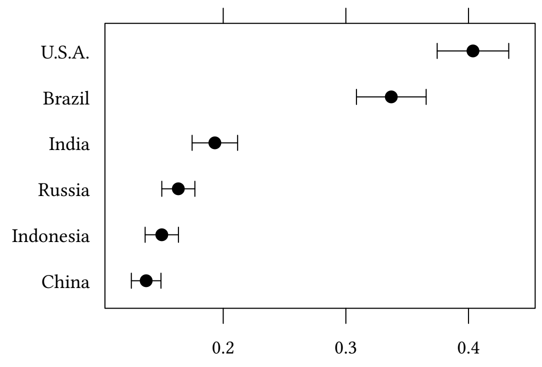
```
latticeExtra::segplot(reorder(factor(country),mean) ~ min+max,centers=mean,
        xlab="investment share",draw.bands=FALSE,data=pwSub2)
```



Segment plots and regression results We can also use segment plots to show regression results. In the following example we use the `pwt6.3` dataset to study the relation between `openc` and `cgdp` per country:

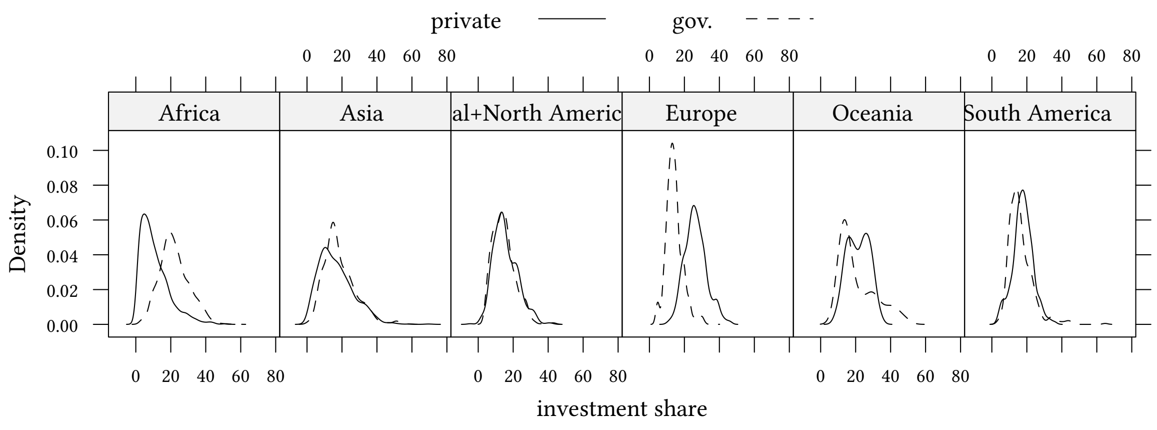
```
reg<-with(pwSub.pw,lm(log(cgdp) ~ openc:country - 1))
reg.ci<-data.frame(cbind(coef(reg),confint(reg)))
names(reg.ci)<-c("coef","lower","upper")
reg.ci[["country"]]<-factor(sub("openc:country","",rownames(reg.ci)))
reg.ci

                             coef     lower     upper   country
openc:countryChina      0.1371746 0.1250925 0.1492567     China
openc:countryU.S.A.     0.4036432 0.3744708 0.4328155    U.S.A.
openc:countryIndia      0.1931830 0.1746383 0.2117278     India
openc:countryRussia     0.1634039 0.1499233 0.1768846    Russia
openc:countryIndonesia 0.1499098 0.1363253 0.1634943 Indonesia
openc:countryBrazil     0.3370612 0.3086532 0.3654693    Brazil
```

```
latticeExtra::segplot(reorder(country,coef)~lower+upper,
        centers=coef,data=reg.ci,
        draw.bands=FALSE,
        segments.fun = panel.arrows,
        ends = "both",angle = 90,
        length = 1, unit = "mm")
```



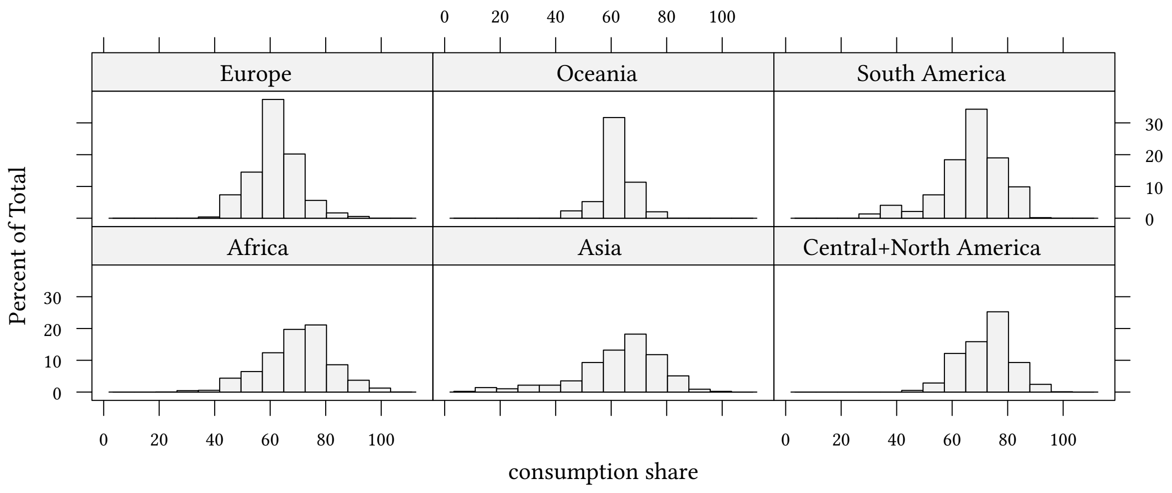
7.4 Densityplots

```
data(pwt5.6)
pwt5.6<-within(pwt5.6,continent<-sub(" & ","+",continent))
keys<-list(text=c("private","gov."),space="top",columns=2,lines=TRUE)
lattice::densityplot(~i+g | continent,data=pwt5.6,plot.points=FALSE,xlab="investment share",
             auto.key=keys)
```



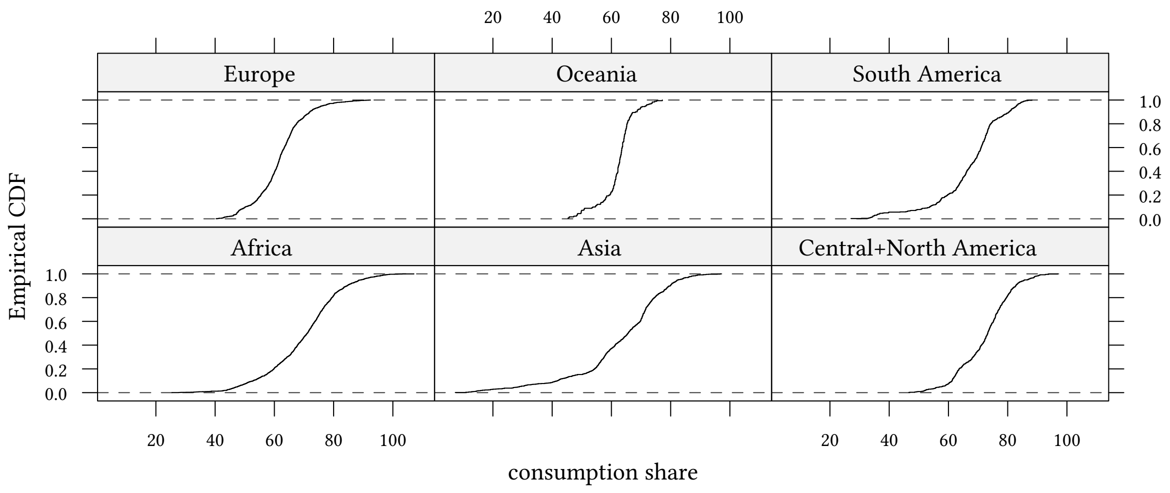
7.5 Histograms

```
lattice::histogram(~c | continent,data=pwt5.6,plot.points=
                FALSE,xlab="consumption share")
```

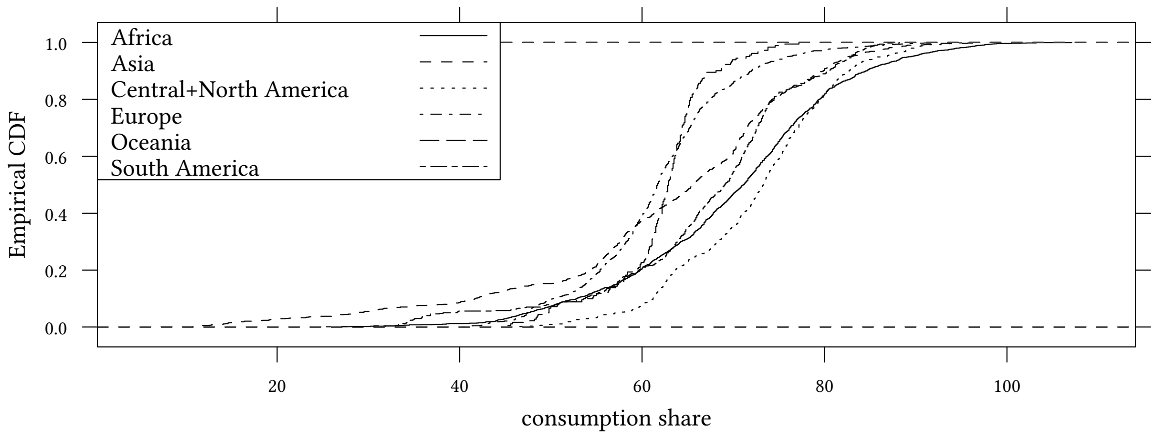


7.6 Empirical cumulative densities

```
library(latticeExtra)
latticeExtra::ecdfplot(~c | continent,data=pwt5.6,xlab="consumption share")
```

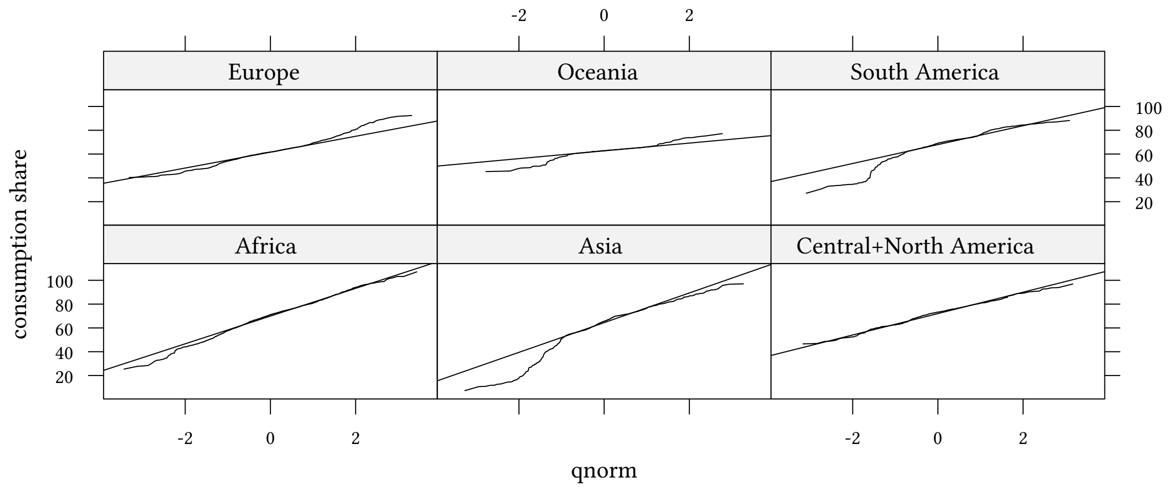


```
key<-list(x=0,y=1,corner=c(0,1),background="white",border=TRUE)
latticeExtra::ecdfplot(~c ,groups= continent,data=pwt5.6,
         auto.key=key,xlab="consumption share")
```

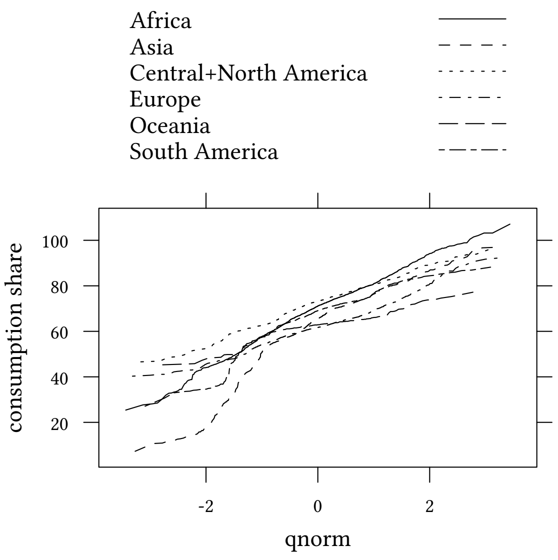


7.7 Q-Q plots

```
lattice::qqmath(~c | continent,data=pwt5.6,ylab="consumption share",type="l",
       panel = function(x, ...) {
           panel.qqmathline(x, ...)
           panel.qqmath(x, ...)
       })
```



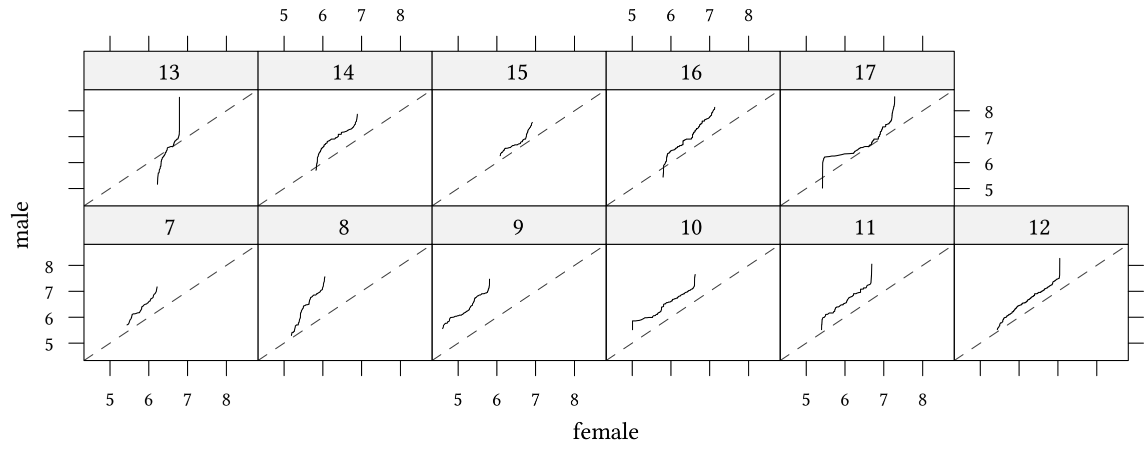
```
lattice::qqmath(~c ,groups= continent,aspect="xy",data=pwt5.6,
       auto.key=list(space="top",
           lines=TRUE,points=FALSE),
       ylab="consumption share",type="l")
```



7.8 Sample Q-Q plots

Here we have to `factor` `ed` to show the values of `ed` in the shingles.

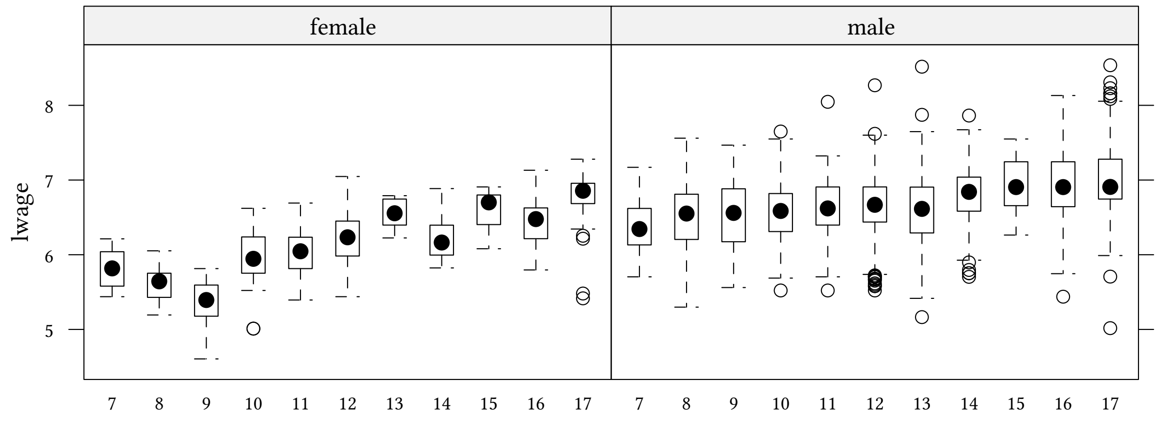
```
library(Ecdat)
data(Wages)
lattice::qq(sex ~ lwage | factor(ed),data=subset(Wages,ed>=7),type="l")
```



7.9 Boxplots

Here we have to `factor` `ed` to make clear whether we want boxplots over `ed` or over `lwage`.

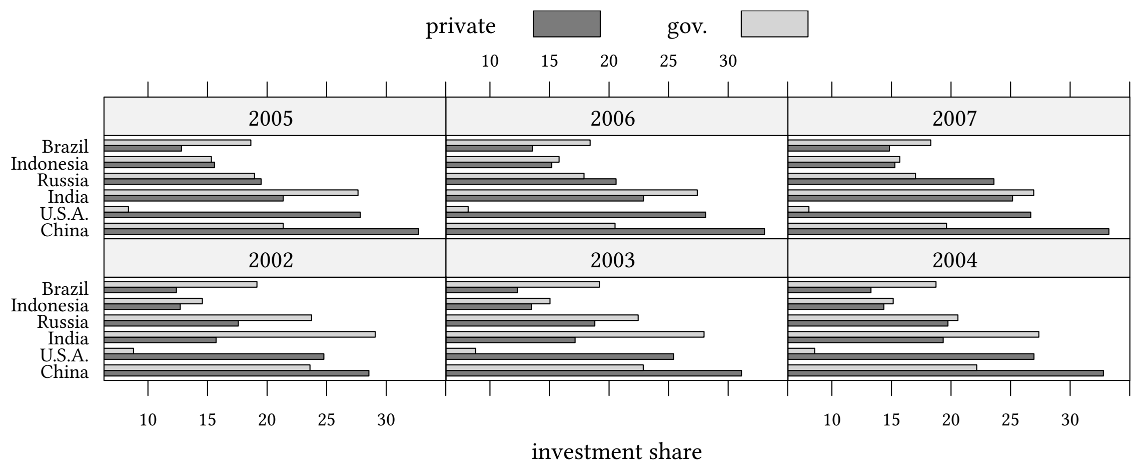
```
lattice::bwplot(lwage  ~ factor(ed) | sex ,data=subset(Wages,ed>=7))
```



7.10 Barcharts

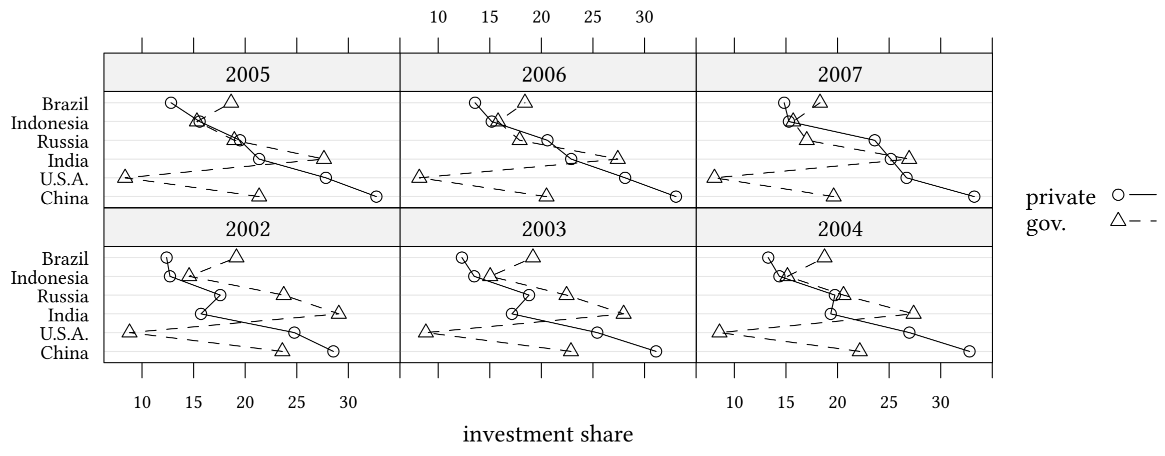
`lattice` can also do bar charts:

```
keys<-list(text=c("private","gov."),space="top",columns=2)
lattice::barchart(country ~ ci+cg| as.factor(year),data=pwSub.pw,xlab="investment share",horizonal=TRUE,
            auto.key=keys)
```



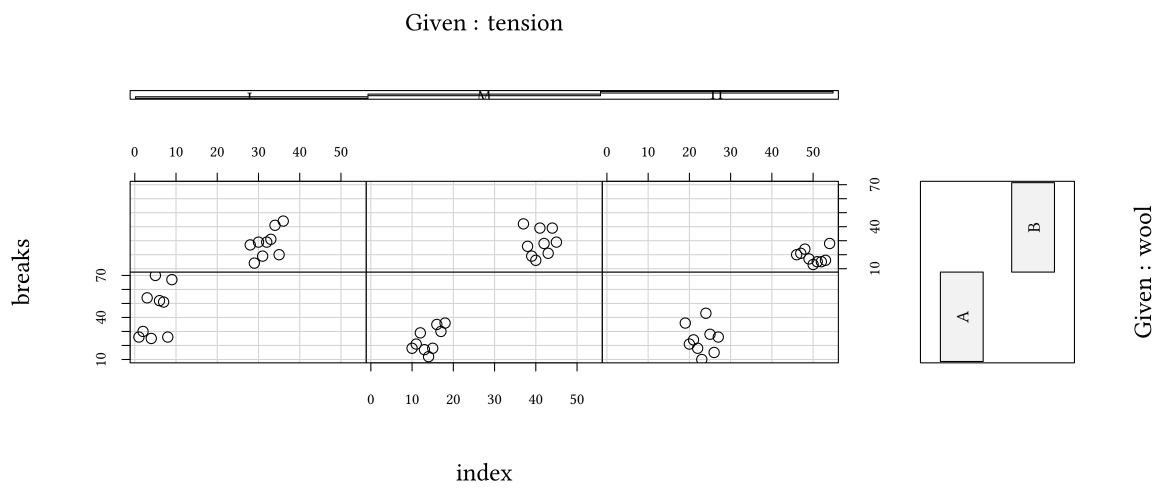
We should note that often a `dotplot` or `xyplot` presents the same data in a better way.

```
keys<-list(text=c("private","gov."),space="right",lines=TRUE,size=2,between=.5)
lattice::dotplot(country ~ ci+cg| factor(year),data=pwSub.pw,xlab="investment share",
        horizonal=TRUE,auto.key=keys,t="b")
```



7.11 Coplots

```
data(warpbreaks)   ## given two factors
coplot(breaks ~ 1:length(breaks) | tension*wool, data = warpbreaks,
       xlab="index")
```



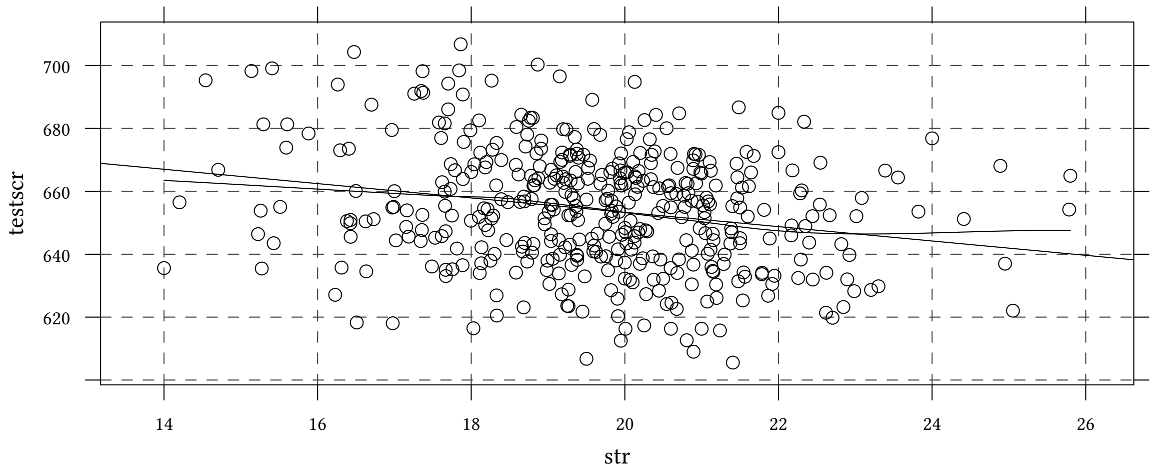
7.12 Parameters

7.12.1 Types

Usually `lattice` renders data as points. The argument `type=(...)` modifies this behaviour. Some useful values are the following:

- `type='p'`: points
- `type='l'`: lines (in the order of the dataset)
- `type='b'`: lines and points
- `type='g'`: a grid
- `type='r'`: a regression line
- `type='smooth'`: a loess smooth line

```
data(Caschool,package="Ecdat")
lattice::xyplot(testscr  ~ str,data=Caschool,type=c("p","g","r","smooth"))
```

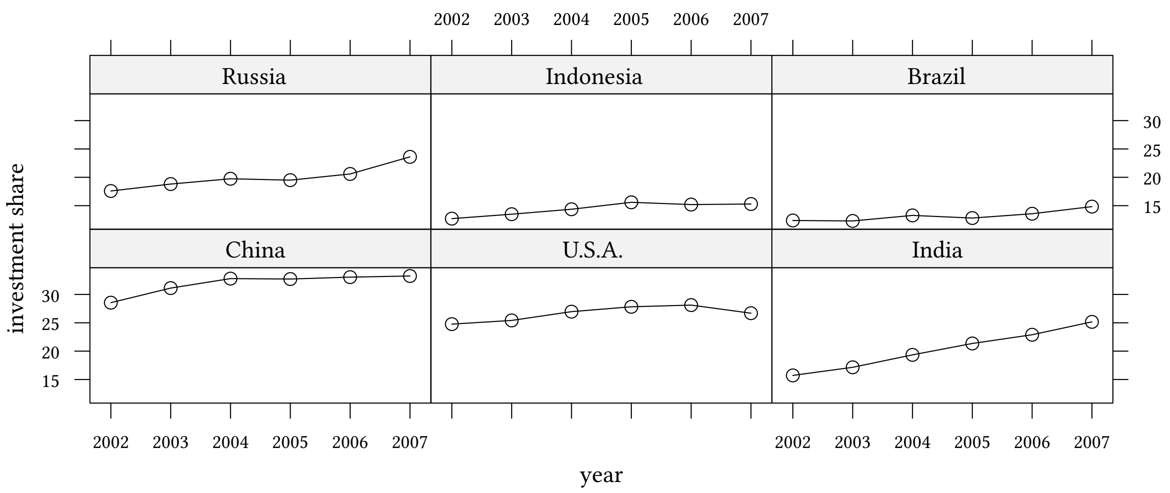


7.12.2 Axes

Different scales for different panels Usually, `lattice` chooses the same scale for all panels in a plot. This can be changed with the help of the parameter `scales`.

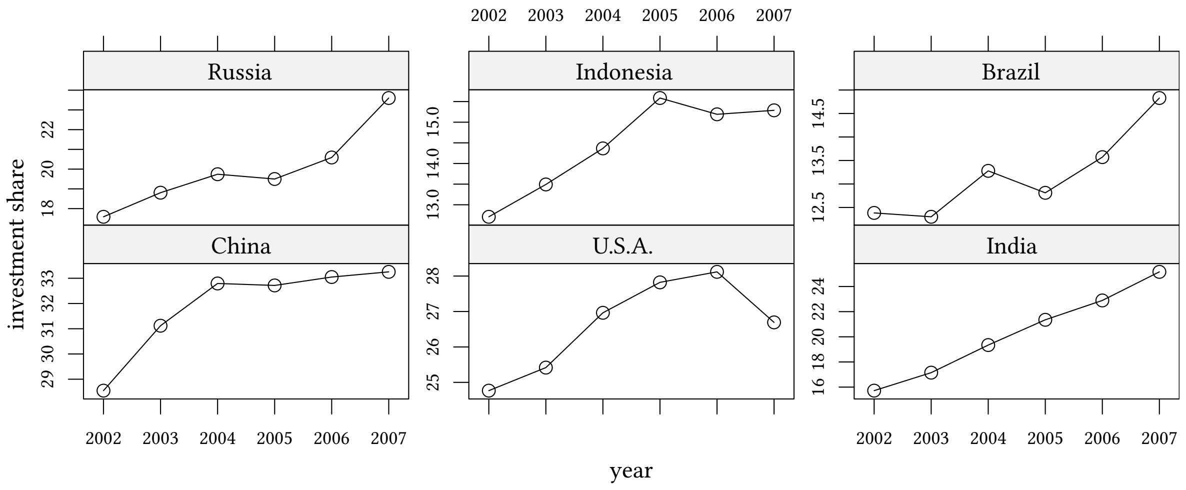
Same scale (the default):

```
lattice::xyplot(ci ~ year| as.factor(country),data=pwSub.pw,ylab="investment share",t="b")
```



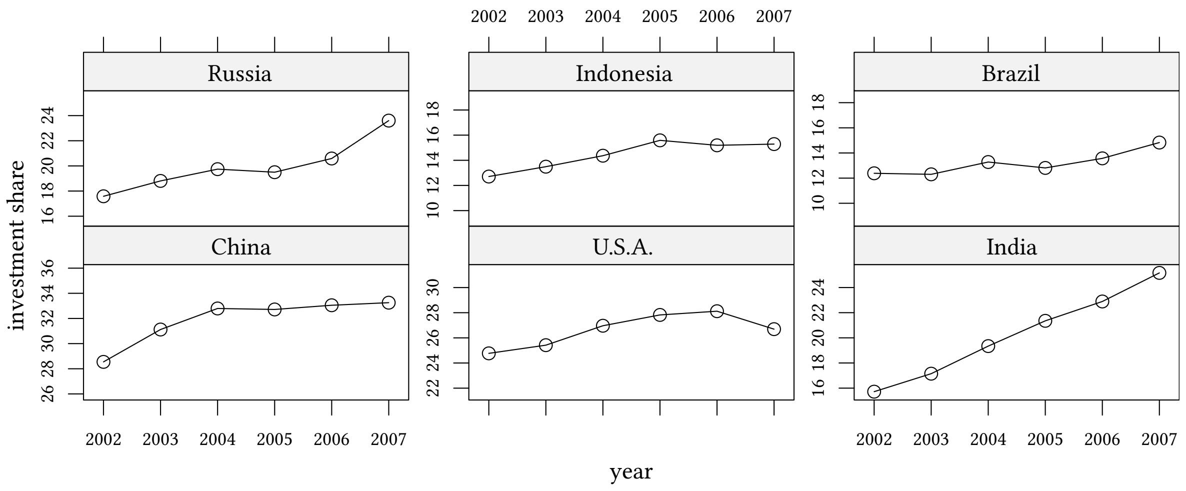
Free scale (`scales=list(x="same",y="free")`):

```
lattice::xyplot(ci ~ year| as.factor(country),data=pwSub.pw,ylab="investment share",t="b",
        scales=list(x="same",y="free"))
```



Sliced scale (`scales=list(x="same",y="sliced")`, scales have the same scale, but different origin):

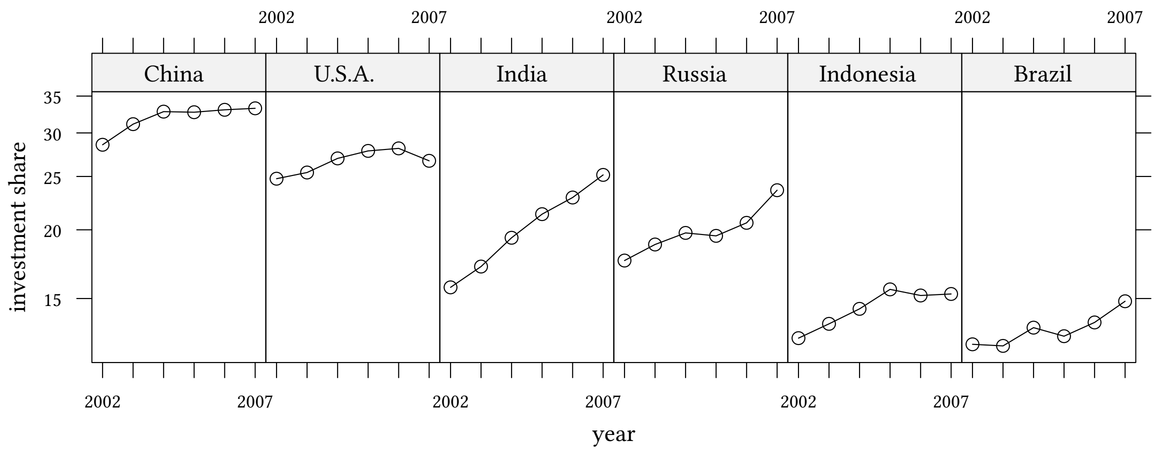
```
lattice::xyplot(ci ~ year| as.factor(country),data=pwSub.pw,ylab="investment share",t="b",
       scales=list(x="same",y="sliced"))
```



Individual axes

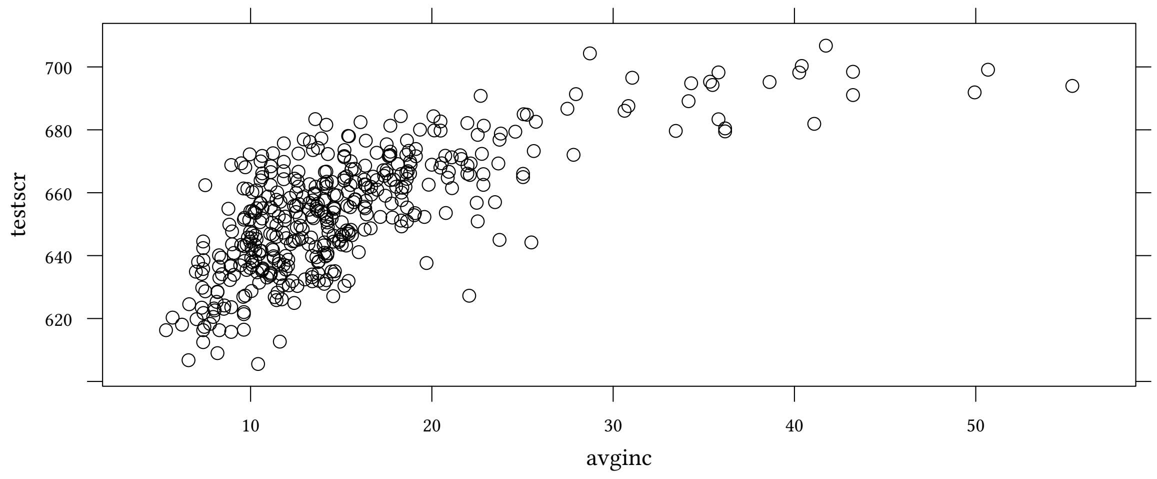
We can influence *where* an axis is labelled as follows:

```
myscale<-list(x=list(at=2002:2007,labels=c(2002,"","","","",2007)),
              y=list(log=TRUE,at=c(15,20,25,30,35)))
lattice::xyplot(ci ~ year| as.factor(country),layout=c(6,1),
       scales=myscale,data=pwSub.pw,ylab="investment share",t="b")
```



More complex plots Let us start with some simple data:

```
lattice::xyplot(testscr ~ avginc,data=Caschool)
```



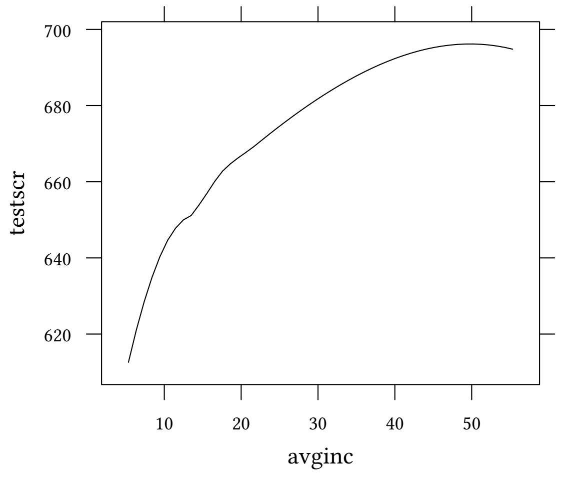
`xyplot` provides a loess smoother, but how can we provide more detail, e.g. confidence bands for the smoother?

Let us first calculate the necessary data:

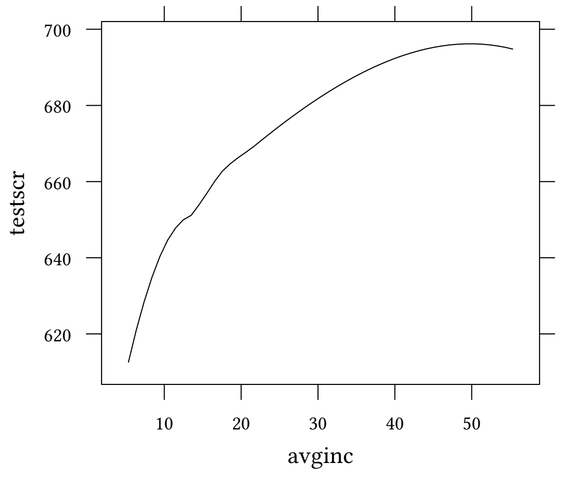
```
data(Caschool,package="Ecdat")
cal.lo<-loess(testscr ~ avginc,data=Caschool)
newx <- with(Caschool,seq(min(avginc),max(avginc),length.out=50))
cal.pred <- predict(cal.lo,newdata=newx,se=TRUE)
cal.df<-with(cal.pred,{data.frame(testscr=fit,
     avginc=newx,
     upper=fit+qnorm(.975)*se.fit,
     lower=fit+qnorm(.025)*se.fit)})
head(cal.df)
```

```
   testscr    avginc    upper    lower
1 612.5926  5.335000 621.0852 604.1000
2 621.0255  6.355265 626.8337 615.2173
3 628.4496  7.375531 632.2356 624.6637
4 634.8632  8.395796 637.3966 632.3298
5 640.2646  9.416061 642.3314 638.1978
6 644.6218 10.436326 646.6465 642.5971
```

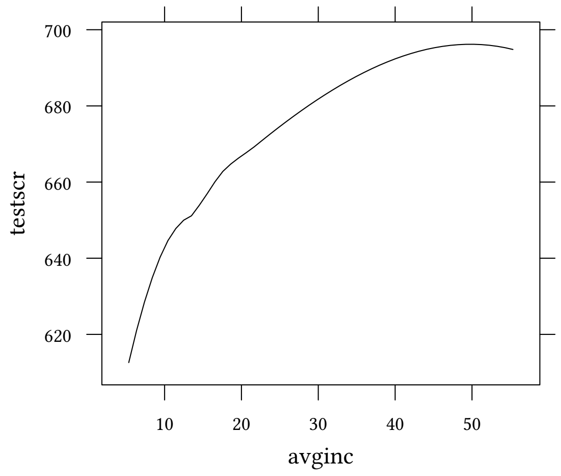
```
lattice::xyplot(testscr ~ avginc,data=cal.df,type="l")
```



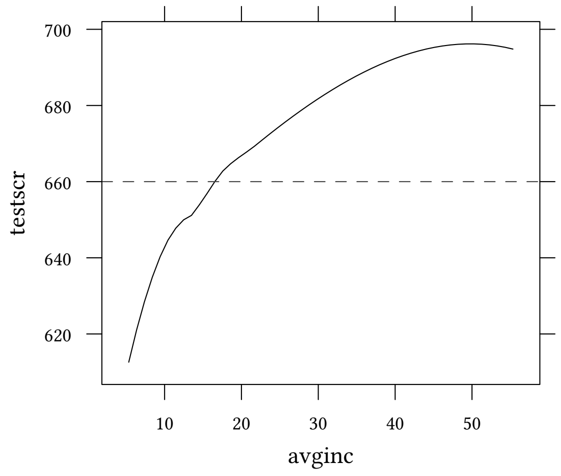
```
lattice::xyplot(testscr ~ avginc,data=cal.df,type="l",
       panel=panel.xyplot)
```



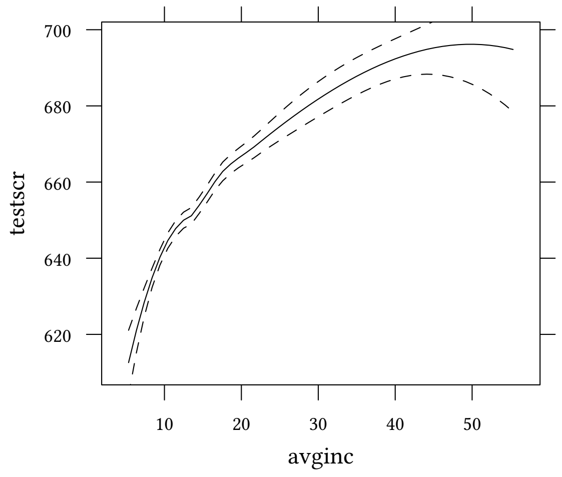
```
lattice::xyplot(testscr ~ avginc,data=cal.df,type="l",
       panel=function(...) panel.xyplot(...))
```



```
lattice::xyplot(testscr ~ avginc,data=cal.df,type="l",
  panel=function(...) {
    panel.xyplot(...);
    panel.refline(h=660)
})
```

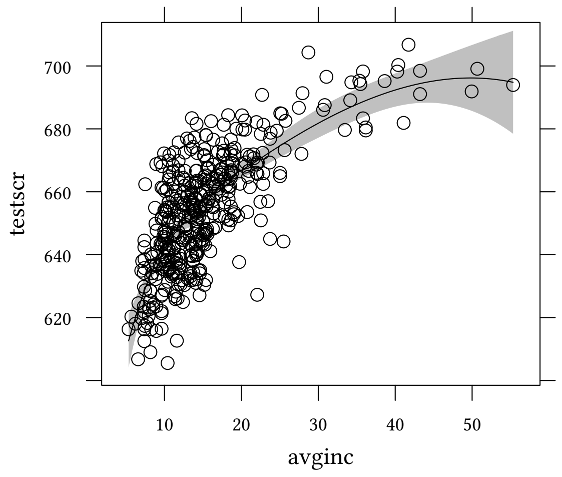


```
with(cal.df,lattice::xyplot(testscr ~ avginc,type="l",
   panel=function(...) {
      panel.xyplot(...);
      panel.xyplot(avginc,upper,type="l",lty=2)
      panel.xyplot(avginc,lower,type="l",lty=2)
}))
```



All this could be done with the help of the built in `panel.smoother` function:

```
lattice::xyplot(testscr ~ avginc,data=Caschool,
   panel=function(...) {
       panel.smoother(...)
       panel.xyplot(...)
   })
```

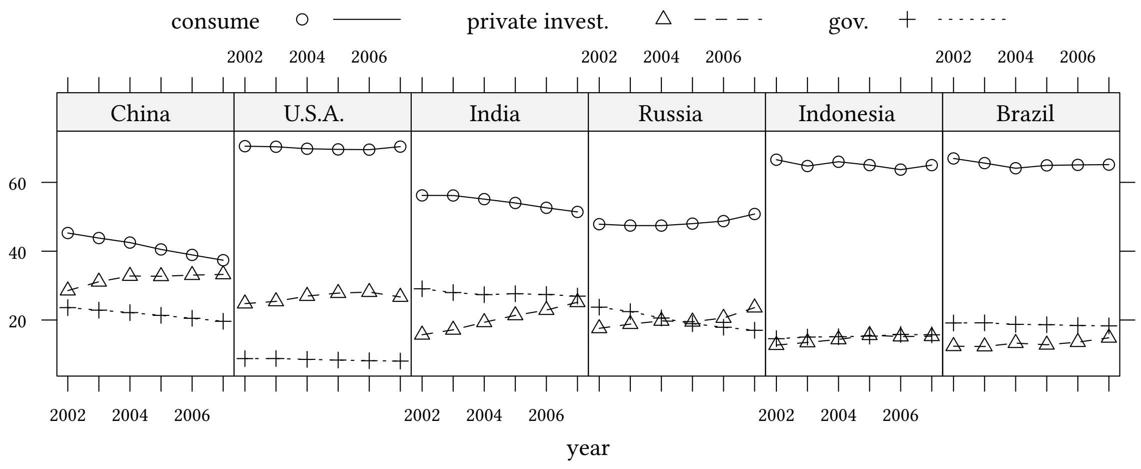


Themes

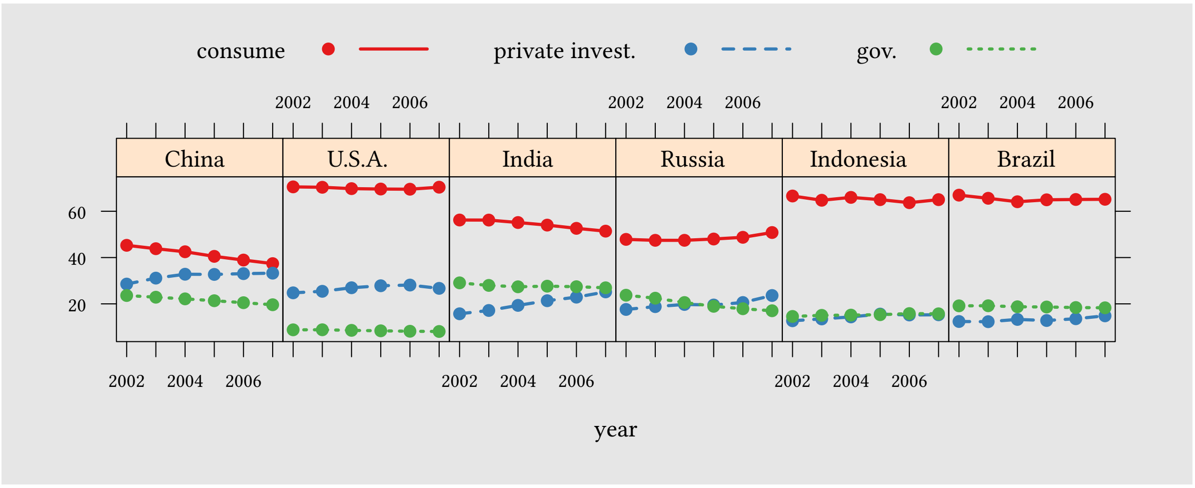
```
keys<-list(text=c("consume","private invest.","gov."),lines=TRUE,
           space="top",columns=3)
mTheme1<-custom.theme(symbol = brewer.pal(3, "Set1"),
  bg = "grey90", fg = "black", pch = 16,lty=1:3,lwd=3)
mTheme2<-custom.theme(symbol = brewer.pal(3, "Pastel1"),
                      fg = "black", lty=1:3,lwd=3)
mTheme3<-custom.theme(symbol = brewer.pal(3, "Paired"),
                      fg = "black")
```

```
mTheme3$strip.background$col=brewer.pal(3, "Pastel2")
pwSub<-lattice::xyplot(cc+ ci + cg ~ year| as.factor(country),layout=c(6,1),
       data=pwSub.pw,ylab="",t="b",auto.key=keys)
```

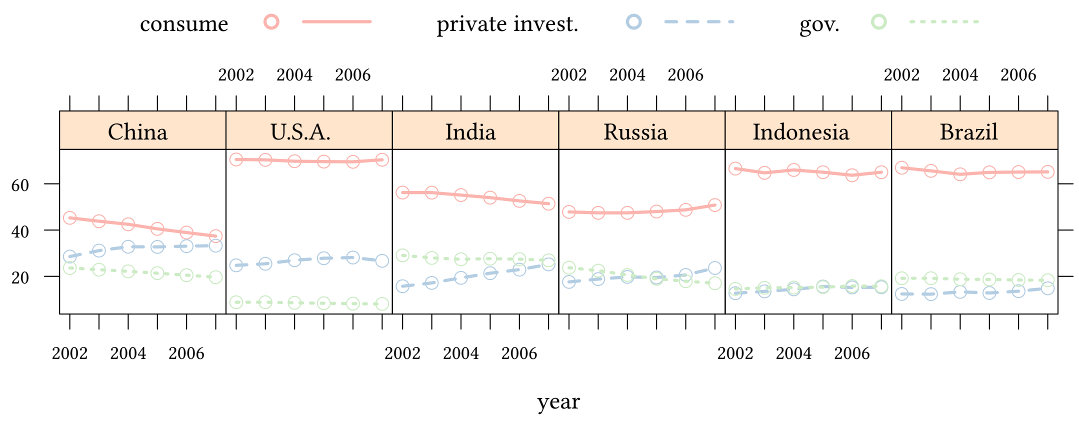
```
pwSub
```



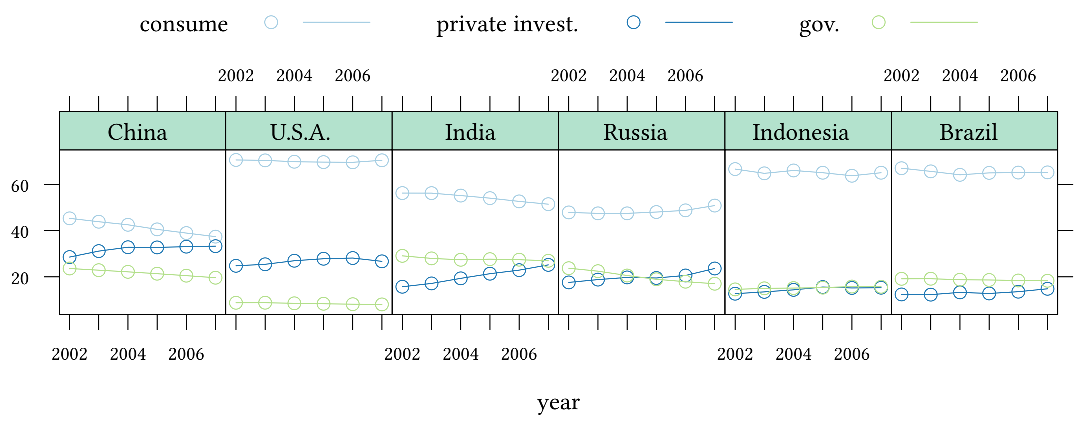
```
update(pwSub,par.settings=mTheme1)
```



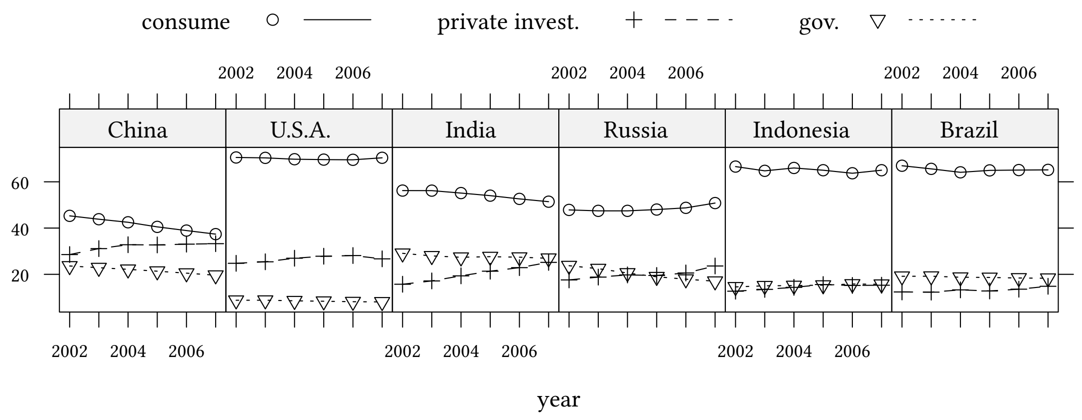
```
update(pwSub,par.settings=mTheme2)
```



```
update(pwSub,par.settings=mTheme3)
```

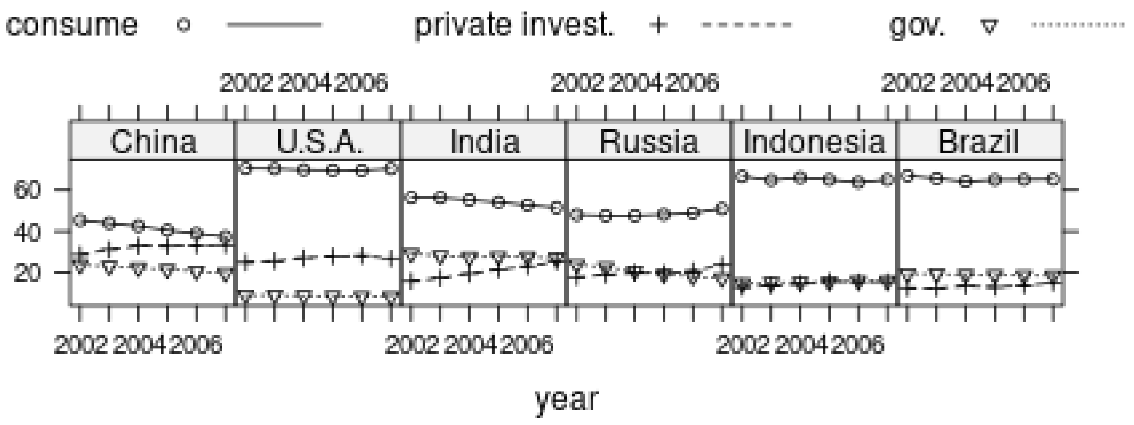


```
update(pwSub,par.settings=standard.theme("pdf", color=FALSE))
```



Vector graphs versus raster images — don't rasterize!:

```
update(pwSub,par.settings=standard.theme("pdf", color=FALSE))
```



Vector graphs

- tikz (for LaTeX)
- eps (sometimes)
- pdf (sometimes)
- svg
- wmf
- …

Raster graphs

- jpeg
- png
- gif
- tiff
- pdf (sometimes)
- …