SpatEcon U0: Prep Script

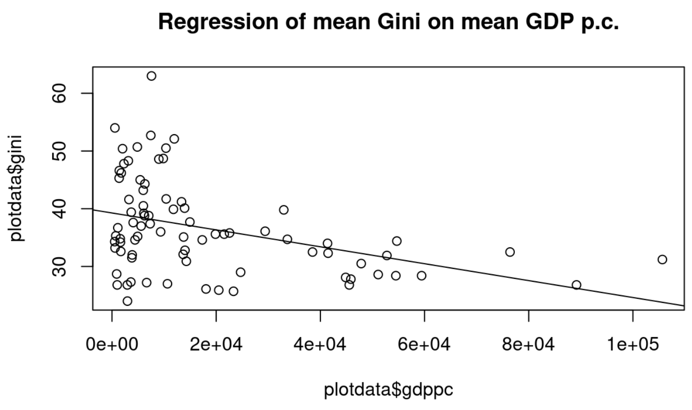
➤ Plotting Data 📈

Before plotting data, you need to tidy up your data. Here I exclude aggregates, only use the year 2014 and exclude NAs.

```
plotdata <- data0[ which(data0$region!="Aggregat
es" & data0$year==2014), ]
plotdata <- na.exclude(plotdata)
```

In R, graphs are typically created interactively.
For example:

```
plot(plotdata$gdppc, plotdata$gini)
abline(lm(plotdata$gini~plotdata$gdppc))
title("Regression of mean Gini on mean GDP p.
c.")
```



The plot() function opens a graph window and plots weight vs. miles per gallon.
The next line of code adds a regression line to this graph.
The final line adds a title.

check the plot function for all it’s arguments, there are many ways to personalize a plot.

```
help(plot)
```

Saving Graphs

You can save the graph via code using one of the following functions:

SpatEcon U0: Prep Script

```
pdf("mygraph.pdf")            #pdf file
win.metafile("mygraph.wmf")   #windows metafile
png("mygraph.png")            #png file
jpeg("mygraph.jpg")           #jpeg file
bmp("mygraph.bmp")            #bmp file
postscript("mygraph.ps")      #postscript file
```

```
pdf("Plot.pdf")
plot(plotdata$gdppc, plotdata$gini)
abline(lm(plotdata$gini~plotdata$gdppc))
title("Regression of mean Gini on mean GDP p.
c.")
dev.off()
```

```
## png
##   2
```

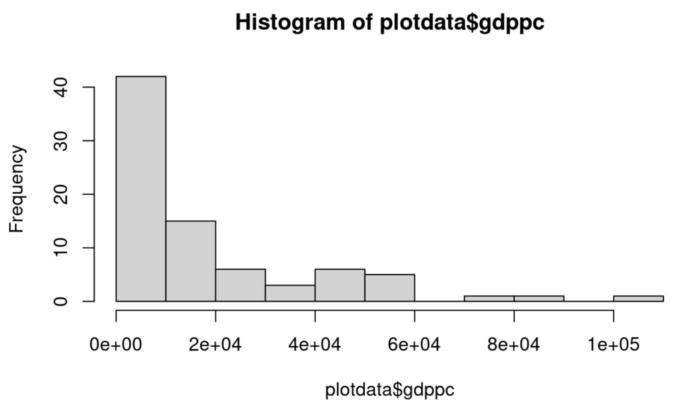
In order to save a plot, we need to use the structure above: open an empty pdf file, write the plot inside and close it again.

Histogram & Density Plot

Histogram

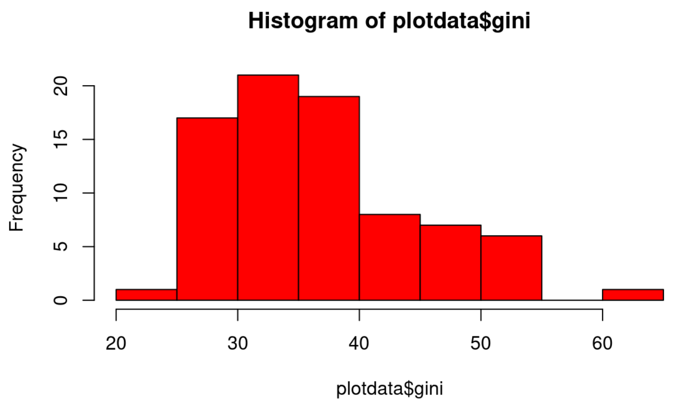
You can create histograms with the function hist(x) where x is a numeric vector of values to be plotted. The option freq=FALSE plots probability densities instead of frequencies.
The option breaks=controls the number of bins.

```
hist(plotdata$gdppc)
# simple histogram
```



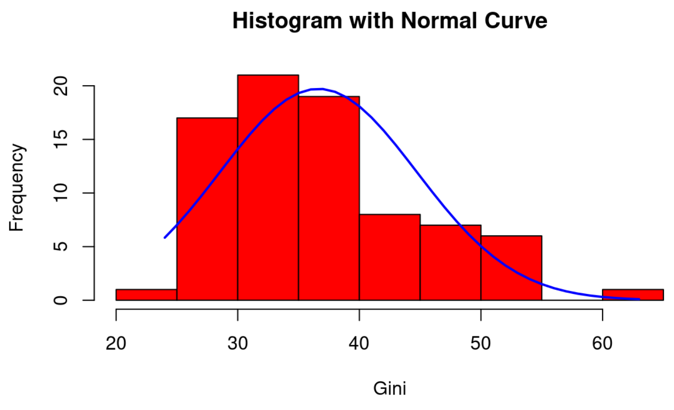
SpatEcon U0: Prep Script

```
hist(plotdata$gini, breaks=10, col="red")
# colored histogram with different number
```



```

# of bins
x <- plotdata$gini
h<-hist(x, breaks=10, col="red", xlab="Gini",
        main="Histogram with Normal Curve")
xfit<-seq(min(x), max(x), length=40)
yfit<-dnorm(xfit, mean=mean(x), sd=sd(x))
# adding a normal curve
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```

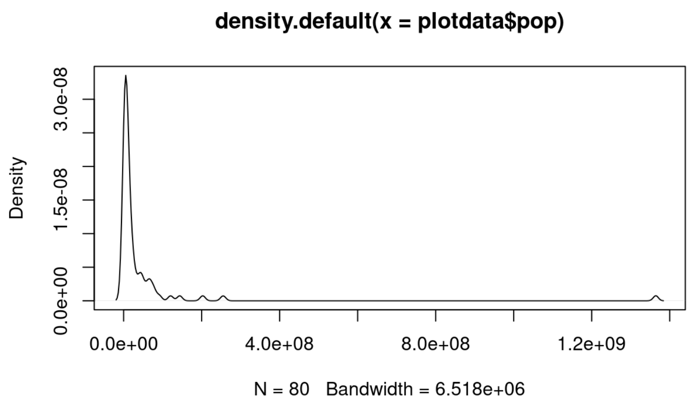


Densityplot

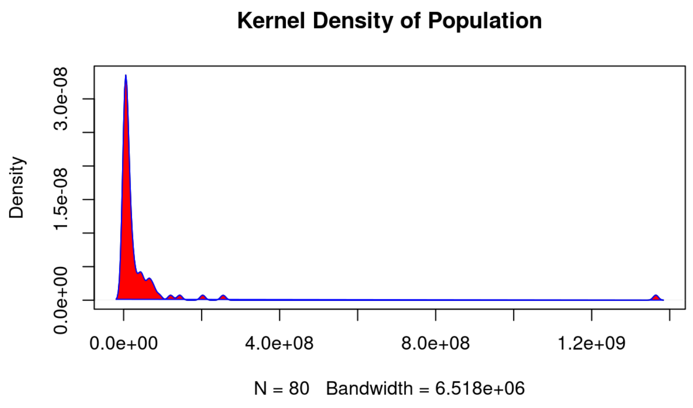
Kernel densityplots are usually a much more effective way to view the distribution of a variable. Create the plot using plot(density(x)) where x is a numeric vector.

```
d <- density(plotdata$pop)
# Kernel densityplot
plot(d)
```

SpatEcon U0: Prep Script



```
plot(d, main="Kernel Density of Population")
# Filled densityplot
polygon(d, col="red", border="blue")
```



Dotplots

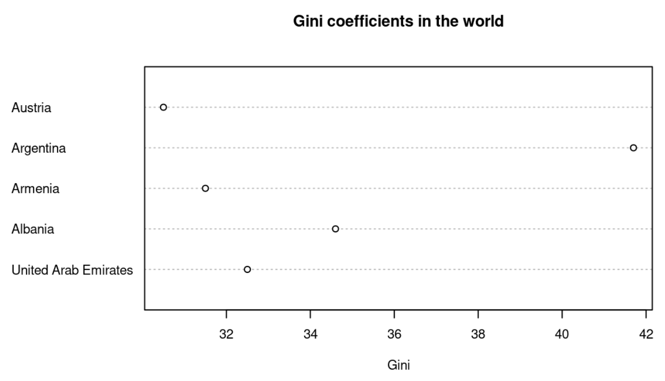
Create dotplots with the dotchart(x, labels=) function, where x is a numeric vector and labels is a vector of labels for each point. You can add a groups=option to designate a factor specifying how the elements of x are grouped. If so, the option gcolor=controls the color of the groups label. cex controls the size of the labels. (Here we use, the short dataset in order to get a nice graph.)

SpatEcon U0: Prep Script

```
plotdatashort <- datashort[ which(datashort$region!="Aggregates" & datashort$year==2014), ]
plotdatashort <- na.exclude(plotdatashort)
# exclude aggregates,

# only 2014

dotchart(plotdatashort$gini, labels=plotdatashort$country, cex=.7,     # simple dotplot
         main="Gini coefficients in the world",
         xlab="Gini")
```

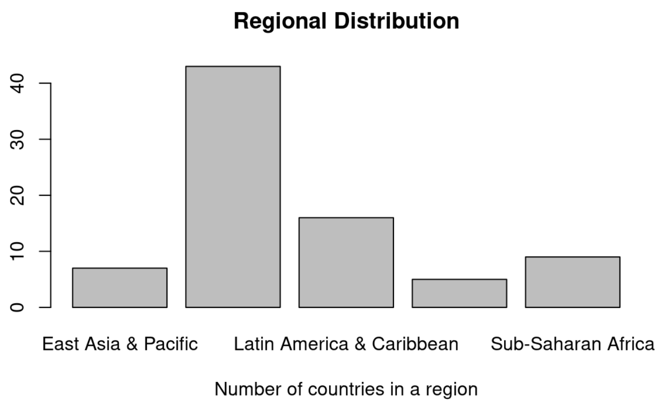


Barplots

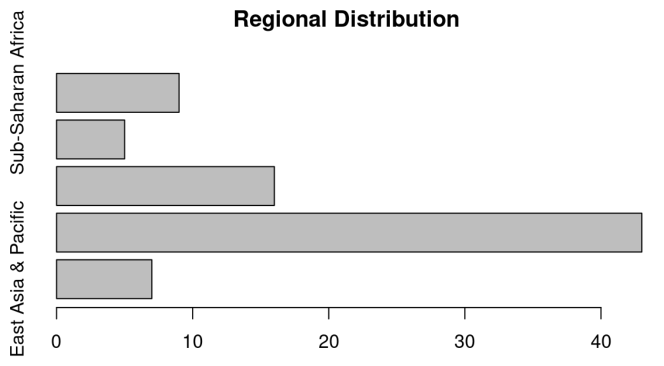
Create barplots with the barplot(height) function, where height is a vector or matrix.
If height is a vector, the values determine the heights of the bars in the plot. If height is a matrix and the option beside=FALSE then each bar of the plot corresponds to a column of height, with the values in the column giving the heights of stacked “sub-bars”. If height is a matrix and beside=TRUE, then the values in each column are juxtaposed rather than stacked.
Include option names.arg=(character vector) to label the bars. The option horiz=TRUE to create a horizontal barplot.

```
counts <- table(plotdata$region)
# simple barplot
barplot(counts, main="Regional Distribution",
        xlab="Number of countries in a region")
```

SpatEcon U0: Prep Script



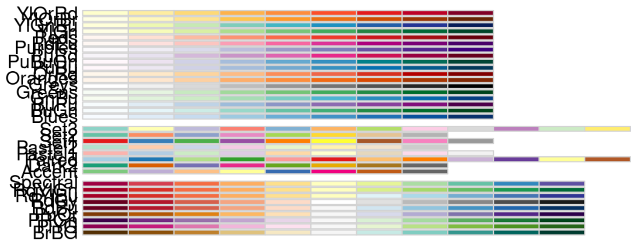
```
counts <- table(plotdata$region)
# simple horizontal barplot with added labels
barplot(counts, main="Regional Distribution", ho
riz=TRUE,
        names.arg=levels(plotdata$region))
```



If we want to use colors in our plots, the package RColorBrewer offers some nicer choices than standard R.

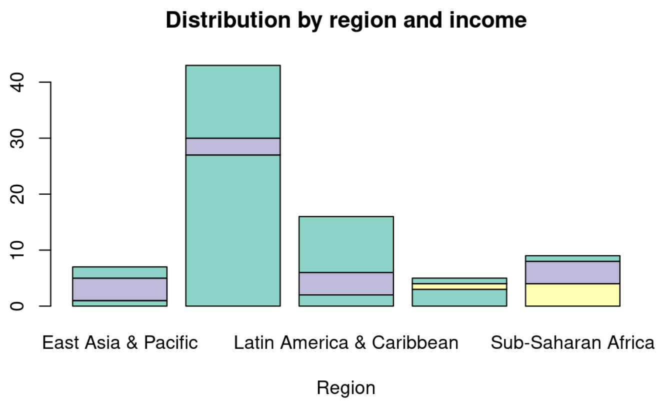
```
# install.packages("RColorBrewer")
library("RColorBrewer")
display.brewer.all()
```

SpatEcon U0: Prep Script



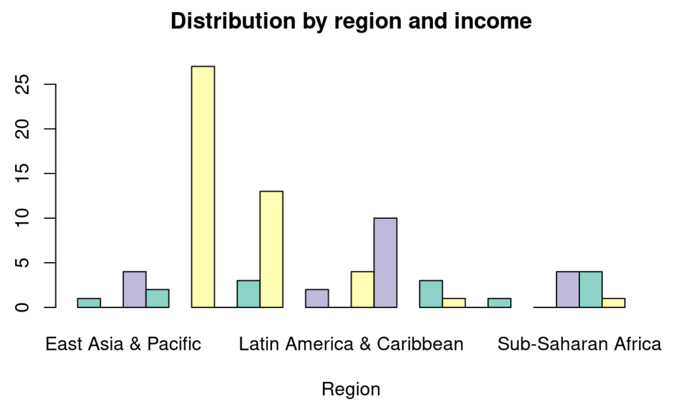
Here are two examples of what barplots could look like:

```
counts <- table(plotdata$income, plotdata$regio
n)            # stacked barplot with colors and
Legend
row.names(counts) <- levels(data0$income)
barplot(counts, main="Distribution by region and
income",
        xlab="Region",
        col=brewer.pal(length(levels(data0$incom
e)), "Set3"), # Note: income level colors out of
palette Set3
        legend = rownames(counts))
```



```
barplot(counts, main="Distribution by region and
income",    # grouped barplot
        xlab="Region",
        col=brewer.pal(length(levels(data0$incom
e)), "Set3"),
        legend = rownames(counts), beside=TRUE)
```

SpatEcon U0: Prep Script



Line Charts

Line charts are created with the function lines(x, y, type=) where x and y are numeric vectors of (x, y) points to connect. type= can take the following values:

```
p       #points
l       #lines
o       #overplotted points and lines
b, c    #points (empty if "c") joined by lines
s, S    #stair steps
h       #histogram-like vertical lines
n       #does not produce any points or lines
```

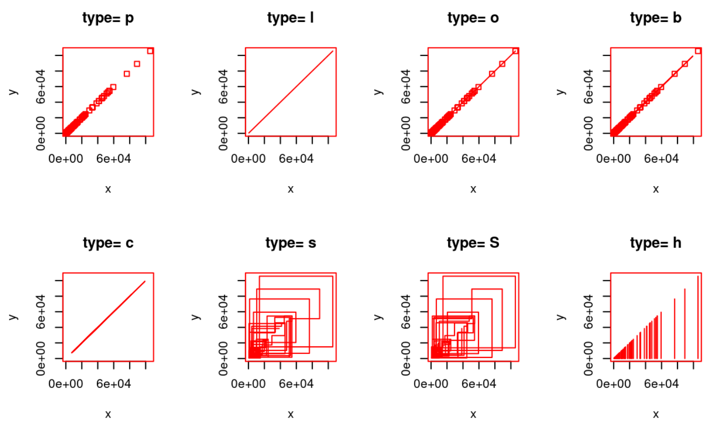
The lines() function adds information to a graph. It can not produce a graph on its own.
Usually it follows a plot(x, y) command that produces a graph.

By default, plot() plots the (x, y) points. Use the type="n" option in the plot() command, to create the graph with axes, titles, etc., but without plotting the points.

For example:
In the following code each of the type=options is applied to the same dataset. The plot() command sets up the graph, but does not plot the points.

SpatEcon U0: Prep Script

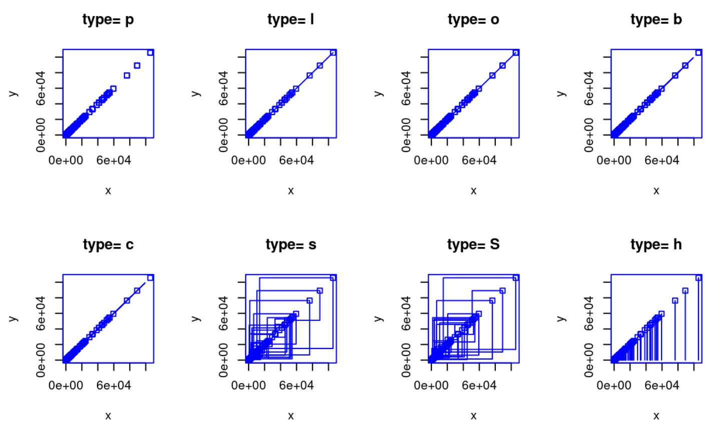
```
x <- plotdata$gdppc; y <- x
# specify data
par(pch=22, col="red")
# plotting symbol and color
par(mfrow=c(2, 4))
# all plots on one page
opts = c("p", "l", "o", "b", "c", "s", "S", "h")
for(i in 1:length(opts)){
  heading = paste("type=", opts[i])
  plot(x, y, type="n", main=heading)
  lines(x, y, type=opts[i])
}
```



Next, we demonstrate each of the type=options when plot( ) sets up the graph and does plot the points.

```
x <- plotdata$gdppc; y <- x
# specify data
par(pch=22, col="blue")
# plotting symbol and color
par(mfrow=c(2, 4))
# all plots on one page
opts = c("p", "l", "o", "b", "c", "s", "S", "h")
for(i in 1:length(opts)){
  heading = paste("type=", opts[i])
  plot(x, y, main=heading)
  lines(x, y, type=opts[i])
}
```

SpatEcon U0: Prep Script



As you can see, the type="c" option only looks different from the type="b" option if the plotting of points is suppressed in the plot() command.

To reset the plot options use:

```
dev.off()
```

```
## null device
##           1
```

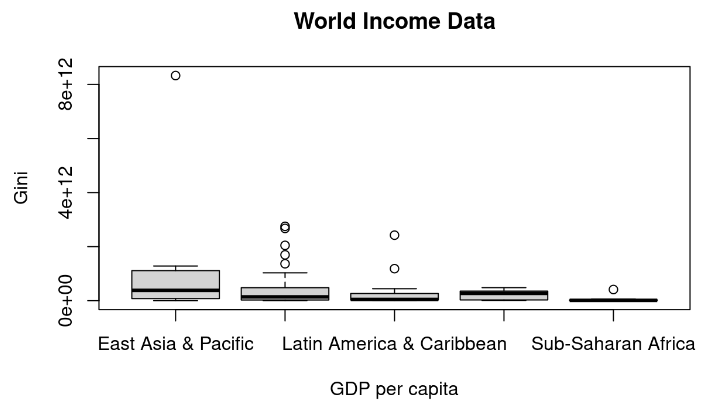
Boxplots

Boxplots can be created for individual variables or for variables by group. The format is boxplot(x, data=), where x is a formula and data= denotes the data frame providing the data. An example of a formula is y~group where a separate boxplot for numeric variable y is generated for each value of group. Add varwidth=TRUE to make boxplot widths proportional to the square root of the samples sizes. Add horizontal=TRUE to reverse the axis orientation.

For example: A Boxplot of GDP by region

```
boxplot(gdp~region, data=plotdata, main="World I
ncome Data",
        xlab="GDP per capita", ylab="Gini")
```

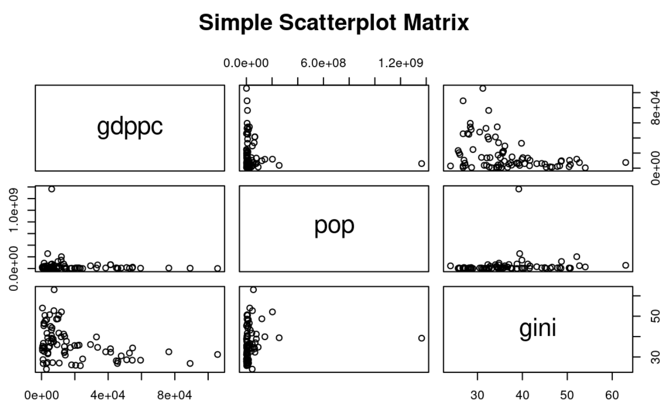
SpatEcon U0: Prep Script



Scatterplots

There are many ways to create a scatterplot in R. The basic function is plot(x, y), where x and y are numeric vectors denoting the (x, y) points to plot. (The "pairs"-plot

SpatEcon U0: Prep Script



Previous Topic