Chapter 2: Visualizing data

Contents

1 Introduction

In this chapter we look at graphical summaries that allow us to visualize the qualitative aspects of the data in order to understand the distribution of the data, the general tendency, and the spread. (Numeric summaries that help us understand how the data is distributed were discussed in the previous chapter on descriptive statistics.)

- **Categorical data** can be summarized by tables and represented graphically with bar plots and pie charts.
- For **continuous data** we mainly rely on histograms and boxplots.

The vocabulary to describe the shape of the distribution plays an important role in the interpretation of these graphical presentations. This includes the concepts of modes and peaks of a distribution, the symmetry or skewness of a distribution and the length of the tails.

2 Categorical variables

2.1 Frequency table

To produce a univariate frequency table for `x`:

```
table()
```

Import the data set *temp_warm.txt* as `temperature`. Create a frequency table for the variable `Area`.

```
area.freq <- table(temperature$Area)
area.freq
```

```
##
##  East North South  West
##     8     8    10     9
```

Or, we can use

```
xtabs(~temperature$Area)
```

```
## temperature$Area
##  East North South  West
##     8     8    10     9
```

Same but **relative frequencies**:

```
area.Rfreq <- table(temperature$Area)/nrow(temperature)
round(area.Rfreq, 2)
```

```
##
##  East North South  West
##  0.23  0.23  0.29  0.26
```

Remark:
More sophisticated table using `gmodels` package

```
install.packages("gmodels")
library(gmodels)
```

Generate table for `Area` variable (from data set `temperature`)

```
CrossTable(temperature$Area, digits = 2, prop.r = FALSE, prop.c = FALSE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##            |      East |     North |     South |      West |
##            |-----------|-----------|-----------|-----------|
##            |         8 |         8 |        10 |         9 |
##            |      0.23 |      0.23 |      0.29 |      0.26 |
##            |-----------|-----------|-----------|-----------|
##
##
##
##
```

2.2 Bar charts

In a **bar chart**, the levels of the variable are arranged in some order and the frequency of each level is represented by a bar with a height proportional to the frequency .

To produce a bar chart:

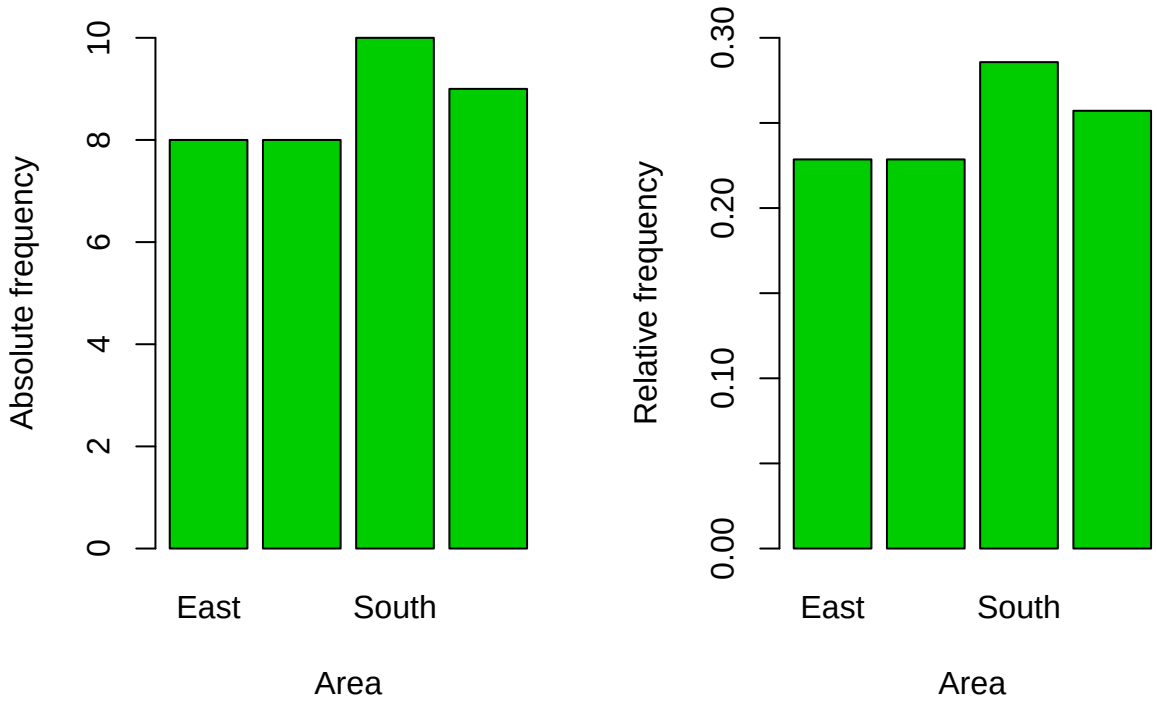
```
barplot()
```

The argument is the frequency (or relative frequency) table.
Some possible arguments:

- `horiz = TRUE` for horizontal bars
- `horiz = FALSE` for vertical bars (default)

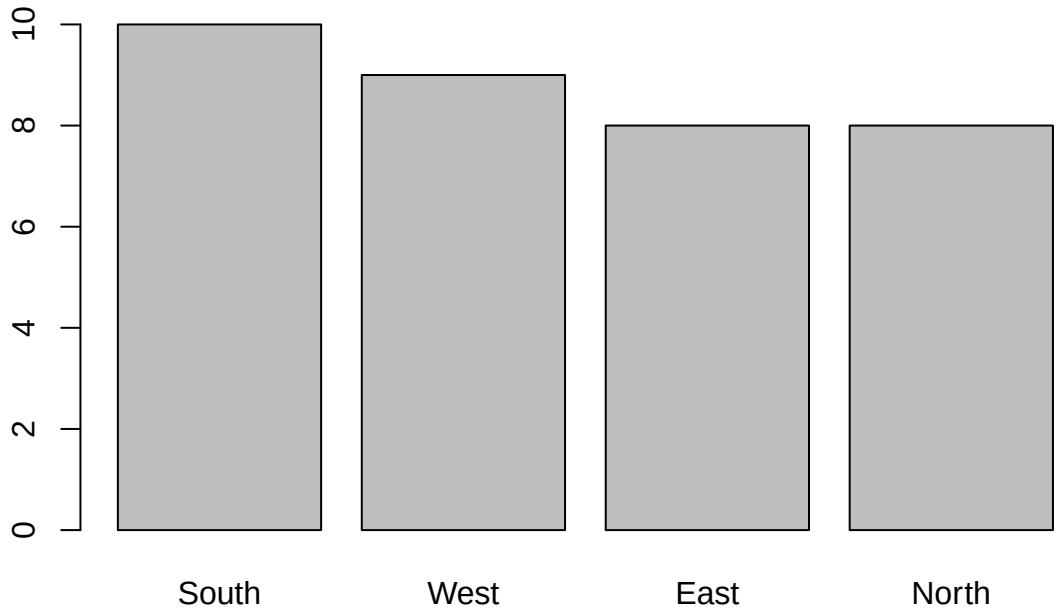
Draw a bar chart for `Area` variable with absolute frequencies and one with relative frequencies

```
area.freq <- table(temperature$Area)
area.Rfreq <- table(temperature$Area)/nrow(temperature)
par(mfrow = c(1,2))
# Draw a bar chart for `Area` variable with absolute frequencies
bp.f <- barplot(area.freq, xlab = "Area", ylab = "Absolute frequency", ylim = c(0,10), col = 3)
# Draw a bar chart for `Area` variable with relative frequencies
bp.Rf <- barplot(area.Rfreq, xlab = "Area", ylab ="Relative frequency", ylim = c(0,0.30),col=3)
```



Remark:
To have the `barplot` sorted by frequency use the function `sort`:

```
barplot(sort(area.freq, decreasing = TRUE))
```



2.3 Pie charts

A **pie chart** is a circular chart divided into segments, illustrating relative magnitudes or frequencies or proportions. It is used for categorical data.

To produce a pie chart for a univariate frequency table:

```
pie()
```

Create a pie chart for the variable `Area` from the data set `temperature`.

```
pp.f <- pie(area.freq, main = "Area", col = rainbow(length(area.freq)))
```

2.4 Dot charts

The default **dot chart** shows the values of variables as big dots on a horizontal display over the range of the data. Differences from the maximum and minimum values are very obvious, but to see their absolute values we must look at the scale.

To produce a dot chart:

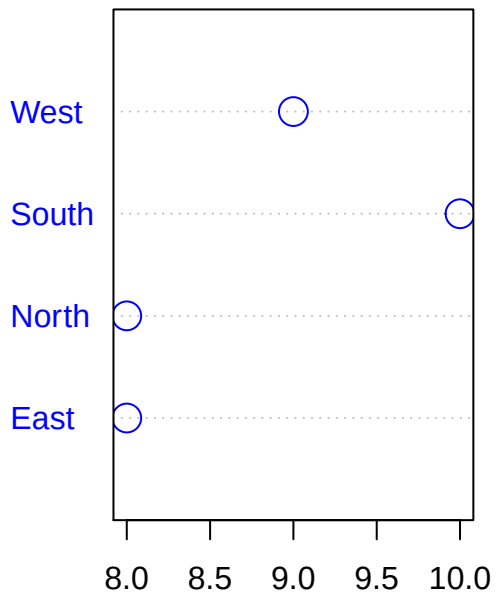
```
dotchart()
```

Create a dot chart for the `Area` variable of the data set `temperature`

```
par(mfrow=c(1,2))

area.freq <- table(temperature$Area) # This is a frequency table
area.df <- data.frame(area.freq)
dp.f <- dotchart(area.df$Freq, labels = area.df$Var1, color = 4, pt.cex = 2)

# Now the same, but with relative frequency
area.Rfreq <- table(temperature$Area)/nrow(temperature)
areaR.df <- data.frame(area.Rfreq)
dp.Rf <- dotchart(areaR.df$Freq, labels = areaR.df$Var1, xlim = c(0.20, 0.30), color=6, pt.cex=2)
```



3 Continuous variables

3.1 Histograms

When dealing with large sample sizes, you can obtain considerable information and get a good overall picture of the situation by grouping the data into several classes or groups. The result is known as a frequency distribution or **histogram**. The frequency distribution provides an easy way to visualize the central tendency, the shape of the distribution and to determine the variability or spread of the data. Frequency distributions also provide a means of comparing observed data with certain known standard patterns called distribution functions.

The histogram uses bars to indicate the frequency or proportion, but for an interval (and not a category which is used in the bar plot). The construction is as follows:

1. First, a set of contiguous of disjoint intervals or *bins* is chosen, covering all data points. Disjoint means no overlap.
2. Next the number of data points or frequency in each of these intervals is counted.
3. Finally a bar is drawn above the interval so that the area of the bar is proportional to the frequency. If the intervals defining the bins all have the same length, then the height of the bar is always proportional to the frequency. If the intervals defining the bins do not have the same length, then the height of the bar is *not necessarily* proportional to the frequency!

To produce a histogram for a vector of values `X`:

```
hist(X)
```

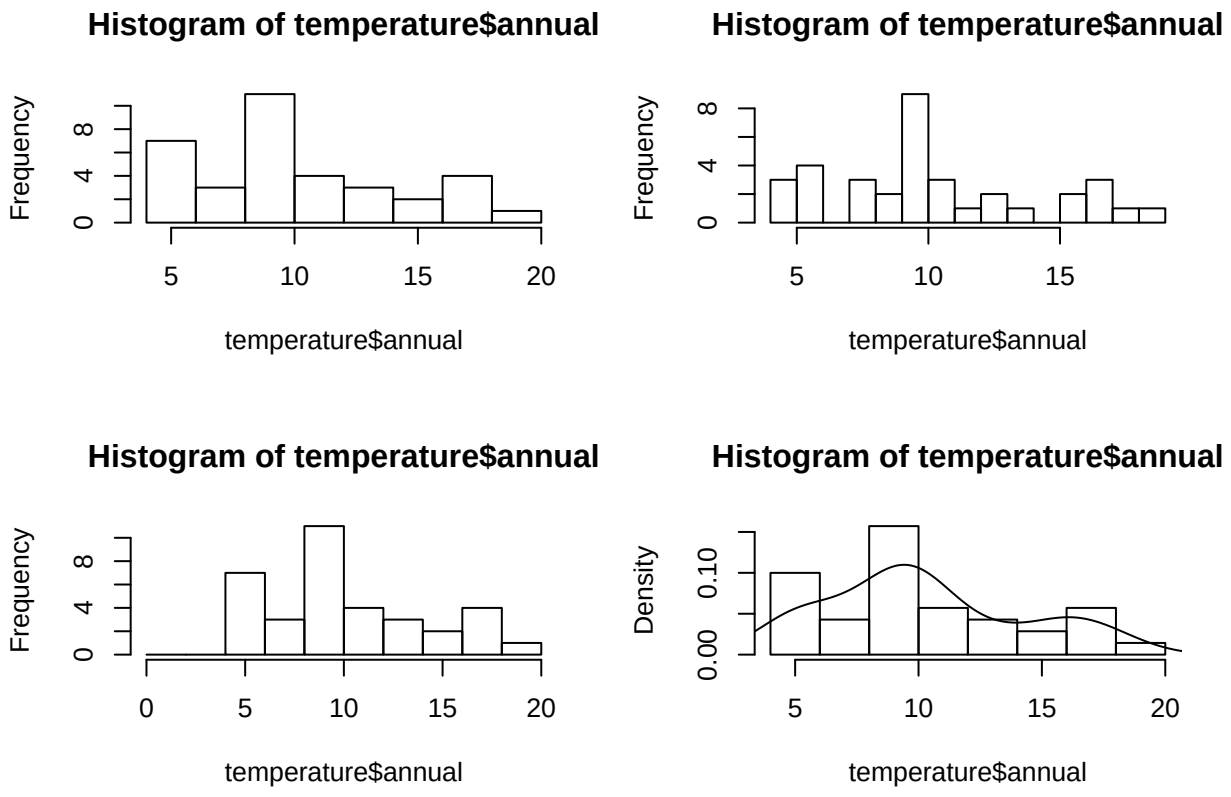
Some possible arguments:

- `main`: plot title
- `col`: change bar colors
- `breaks`: control number of bins and bin sizes
- `probability`: density or frequency histogram
- `xlab`: label for the x-axis
- `ylab`: label for the y-axis

Example with the `temperature` data

```
par(mfrow = c(2,2))

# Basic histogram for "annual" variable
hist(temperature$annual)
# Define no. of bins
hist(temperature$annual, breaks = 10)
# Define specific bins
hist(temperature$annual, breaks = seq(0, 20, 2))
# Use relative frequency
hist(temperature$annual, probability = TRUE)
# Add empirical density curve
lines(density(temperature$annual))
```



3.2 Boxplots

To produce a boxplot for a vector of values `X`:

```
boxplot(X)
```

Some possible arguments:

- `main`: plot title
- `range`: define the range of the whiskers
- `xlab`: label for the X axis
- `ylab`: label for the Y axis

A **boxplot** is a visual representation of the five number summary. It displays the minimal, maximum, median and the quartiles. The spread is indicated by a box of which the length corresponds to the IQR. The range is shown with two whiskers. In the simplest case these correspond to the minimum and maximum values. If any outliers are thought or known to be present in the data, these can be marked as separate points (and thus fall outside of the whiskers!). By default R uses another convention where the length of the whiskers is no longer than 1.5 times the length of the box. Data values not contained in this range are then marked as separate points.

Symmetry of the distribution is reflected in symmetry of the box plot in both the location of the median in the box and the length of the two whiskers.

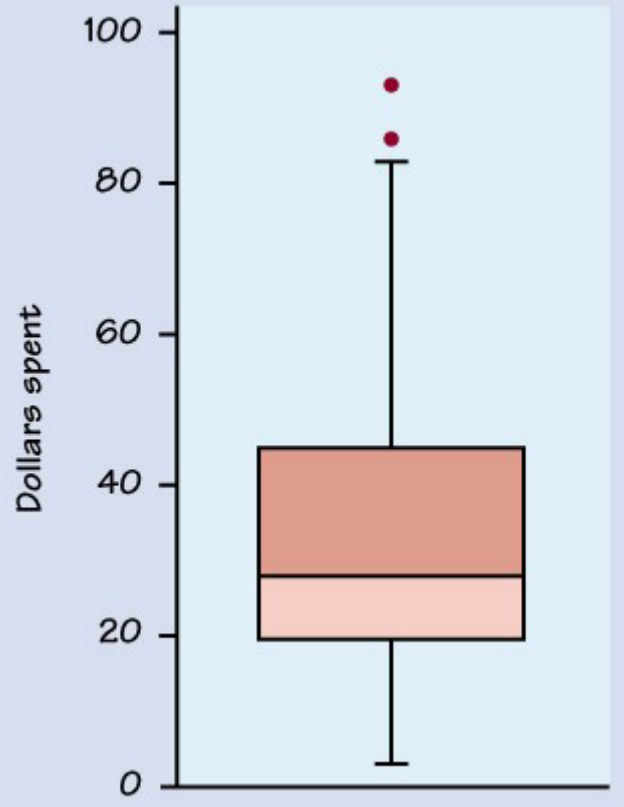
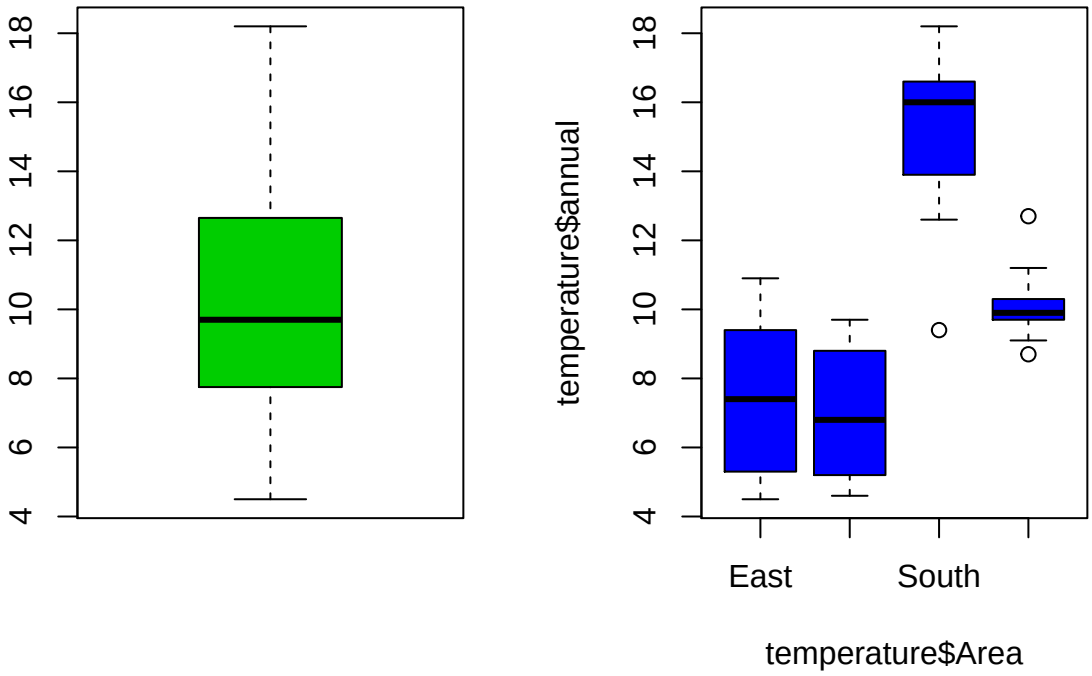


Figure 1: *Example of a boxplot*

Create a boxplot for the variable `annual`

```
par(mfrow = c(1,2))
# Basic boxplot
boxplot(temperature$annual, col= 11)
# Grouped boxplots
boxplot(temperature$annual ~ temperature$Area, col = 12)
```



4 Scatterplots

There are many scientific relations between numeric variables, for instance pressure is proportional to the temperature. Many relations are not precisely known, prompting an examination of the data. If a bivariate dataset has a natural pairing such as $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ then it makes sense to investigate the data jointly. A **scatterplot** or scatter graph is a graph to visually display and compare two sets of quantitative or numerical data by displaying them as a set of finite points each having a coordinate on the horizontal X and the vertical Y axis. Scatterplots are useful to detect potential relationships between 2 or more variables. It plots the values of one vector against the other as datapoints.

To create a two-dimensional scatterplot of `X` versus `Y`:

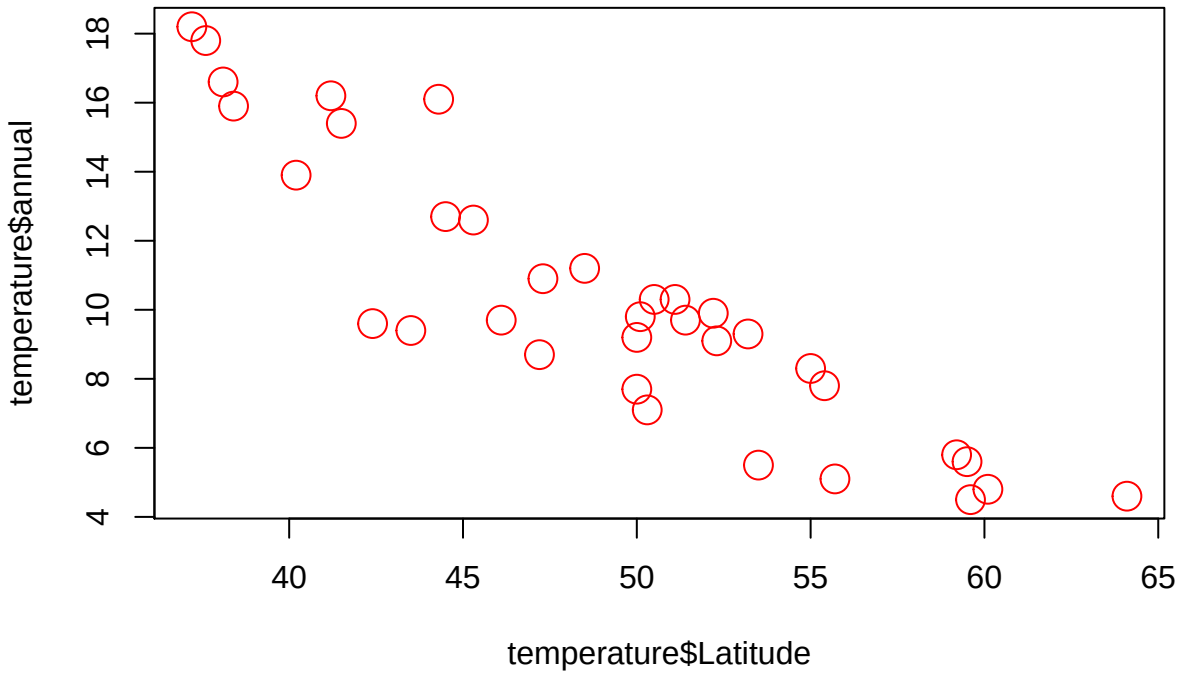
`plot(X, Y)` or `plot(Y ~ X)`

Some possible arguments:

- `main`: plot title
- `xlab`: label for the X axis
- `ylab`: label for the Y axis
- `col`: color of what is plotted
- `xlim`: limits for the X axis
- `ylim`: limits for the Y axis
- `cex`: magnification factor
- `type`: `"p"` for points, `"l"` for lines, `"h"` for histogram like vertical lines, ...
- `pch`: style of the points
- `lty`: type of the line
- `lwd`: thickness of the line

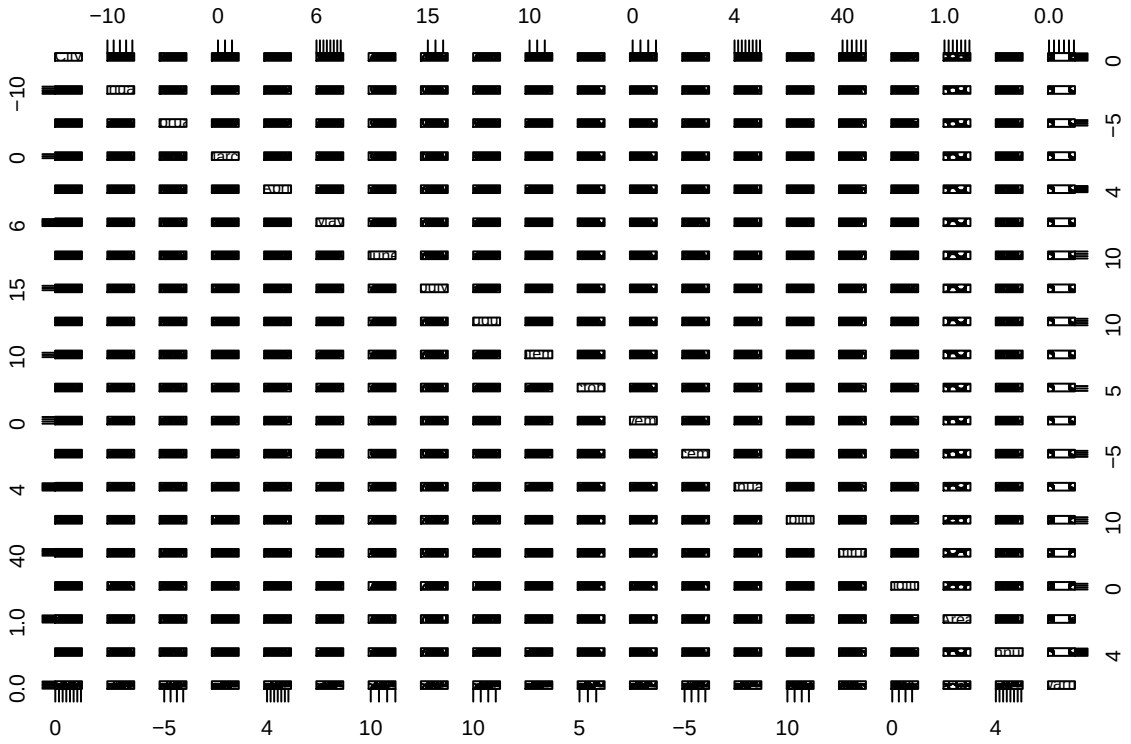
Scatterplot for `annual` and `Latitude` variables

```
plot(temperature$annual ~ temperature$Latitude,col=2, cex=2)
```



All bivariate scatterplots for data set `temperature`

```
plot(temperature)
```



5 Some general R commands for plotting

Various plotting functions for creating and adding to figures:

- `plot()`: Used for scatterplots. It plot points by default. To plot lines, use argument `type = "l"`.
- `points()`: Draws points
- `lines()`: Similar to `points()` but connects points by lines.
- `abline()`: Function for adding lines to figures. The arguments `a = ...` and `b = ...` will plot the line $ax + b$, the arguments `h = ...` and `v = ...` will plot horizontal and vertical lines.
- `curve()`: Plots function for adding the graph of a function of x. When `add = TRUE`, the graph will be drawn on the current figure using the current range of x values. When `add = TRUE` is not given, a new graph will be produced with range from $= \ldots$ (default 0) to $= \ldots$ (default 1). The function to be graphed maybe given by name or written as a function of x.
- `arrows()`: Adds arrows to the figure.
- `text()`: Adds text to figure at specified points
- `title()`: Adds labels to a figure, arguments is `main = ...` for the main title, `sub = ...` for the subtitle, `xlab = ...` for setting the x label and `ylab = ...` for setting the y label.
- `legend()`: Adds a legend to the figure.