Introduction to R

Basic Graphics

Jan-Philipp Kolb

05 März, 2020

FIRST THINGS TO DO

> Don’t try to kiss your data on the first date; rather, you just want to get to know the data:

1. Import the data
2. Review the codebook
3. Learn about the data
4. **Quick visual understanding of the data**

A plot says more than 1000 words

Statements on graphs in R

- Graphical data analysis is great
- Good plots can contribute to a better understanding
- Generating a plot is easy
- Making a good plot can take very long
- Generating plots with R is fun
- Plots created with R have high quality
- Almost every plot type is supported by R
- A large number of export formats are available in R

Graphic types


Violin


Density


Histogram


Boxplot


Ridgeline

Correlation


Scatter


Heatmap


Correlogram


Bubble


Connected scatter


Density 2d

Ranking


Barplot


Spider / Radar

Wordcloud


Parallel


Lollipop


Circular Barplot

Task View for graphics

CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

Maintainer: Nicholas Lewin-Koh
Contact: nikko at hailmail.net
Version: 2015-01-07
URL: https://CRAN.R-project.org/view=Graphics

R is rich with facilities for creating and developing interesting graphics. Base R contains functionality for many plot types including coplots, mosaic plots, biplots, and the list goes on. There are devices such as postscript, png, jpeg and pdf for outputting graphics as well as device drivers for all platforms running R. lattice and grid are supplied with R's recommended packages and are included in every binary distribution. lattice is an R implementation of William Cleveland's trellis graphics, while grid defines a much more flexible graphics environment than the base R graphics.

R's base graphics are implemented in the same way as in the S3 system developed by Becker, Chambers, and Wilks. There is a static device, which is treated as a static canvas and objects are drawn on the device through R plotting commands. The device has a set of global parameters such as margins and layouts which can be manipulated by the user using `par()` commands. The R graphics engine does not maintain a user visible graphics list, and there is no system of double buffering, so objects cannot be easily edited without redrawing a whole plot. This situation may change in R 2.7.x, where developers are working on double buffering for R devices. Even so, the base R graphics can produce many plots with extremely fine graphics in many specialized instances.

One can quickly run into trouble with R's base graphic system if one wants to design complex layouts where scaling is maintained properly on resizing, nested graphs are desired or more interactivity is needed. grid was designed by Paul Murrell to overcome some of these limitations and as a result packages like lattice, ggplot2, vcd or hexbin use grid for the underlying primitives. When using plots designed with grid one needs to keep in mind that grid is based on a system of viewports and graphic objects. To add objects one needs to use grid commands, e.g., `grid.polygon()` rather than `polygon()`. Also grid maintains a stack of viewports from the device and one needs to make sure the desired viewport is at the top of the stack. There is a great deal of explanatory documentation included with grid as vignettes.

The graphics packages in R can be organized roughly into the following topics, which range from the more user oriented at the top to the more developer oriented at the bottom. The categories are not mutually exclusive but are for the convenience of presentation:

https://cran.r-project.org/web/views/Graphics.html

The example dataset

```
install.packages("AmesHousing")
```

```
ames_df <- AmesHousing::make_ames()
```

Variables used in this section

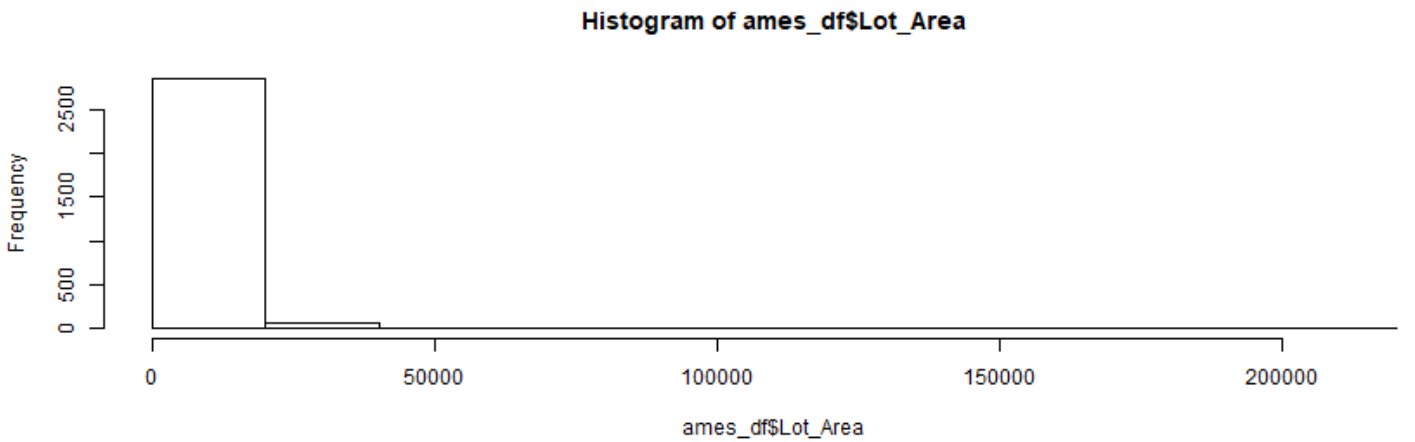
- `Lot_Area`: Lot size in square feet
- `Alley`: Type of alley access to property
- `Street`: Type of road access to property

Histogram - The `hist()` function

We create a histogram of the variable `duration`:

```
?hist
```

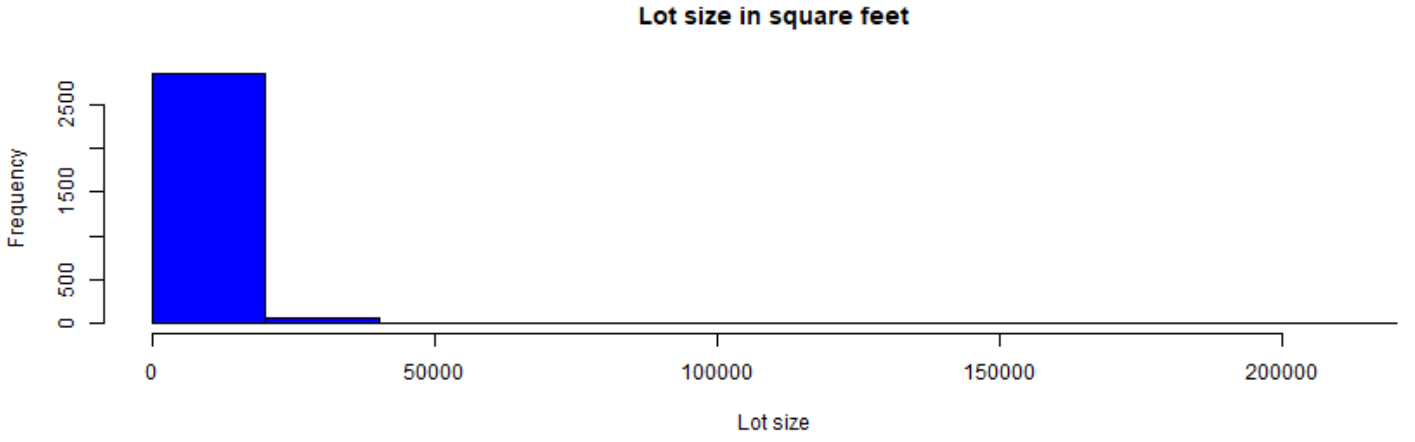
```
hist(ames_df$Lot_Area)
```



Histogram

- Command `hist()` plots a histogram
- At least one observation vector must be passed to the function
- `hist()` has many more arguments, which all have (meaningful) default values

```
hist(ames_df$Lot_Area,col="blue",
     main="Lot size in square feet",ylab="Frequency",
     xlab="Lot size")
```



Further arguments:

- Many of the arguments are valid for all base graphics like `main` or $xlab$`.
- You can see many of them if you call help on `?par`.

```
?plot
# or
?par
```

Graphical Parameters

`adj`

The value of `adj` determines the way in which text strings are justified in `text`, `mtext` and `title`. A value of `0` produces left-justified text, `0.5` (the default) centered text and `1` right-justified text. (Any value in *[0, 1]* is allowed, and on most devices values outside that interval will also work.)

Note that the `adj` *argument* of `text` also allows `adj = c(x, y)` for different adjustment in x- and y- directions. Note that whereas for `text` it refers to positioning of text about a point, for `mtext` and `title` it controls placement within the plot or device region.

`ann`

If set to `FALSE`, high-level plotting functions calling `plot.default` do not annotate the plots they produce with axis titles and overall titles. The default is to do annotation.

`ask`

logical. If `TRUE` (and the `R` session is interactive) the user is asked for input, before a new figure is drawn. As this applies to the device, it also affects output by packages **grid** and **lattice**. It can be set even on non-screen devices but may have no effect there.

This not really a graphics parameter, and its use is deprecated in favour of `devAskNewPage`.

Links

- **R graph gallery**
- **Bioconductor R manual** with an extensive part on graphics
- **Shiny app** for **interactive plot editing**