Data visualization in R

Mikhail Dozmorov

Fall 2016
Why visualize data?

- Four groups
- 11 observations \((x, y)\) per group

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean of (x) in each case</td>
<td>9 (exact)</td>
</tr>
<tr>
<td>Sample variance of (x) in each case</td>
<td>11 (exact)</td>
</tr>
<tr>
<td>Mean of (y) in each case</td>
<td>7.50 (to 2 decimal places)</td>
</tr>
<tr>
<td>Sample variance of (y) in each case</td>
<td>4.122 or 4.127 (to 3 decimal places)</td>
</tr>
<tr>
<td>Correlation between (x) and (y) in each case</td>
<td>0.816 (to 3 decimal places)</td>
</tr>
<tr>
<td>Linear regression line in each case</td>
<td>(y = 3.00 + 0.500x) (to 2 and 3 decimal places, respectively)</td>
</tr>
</tbody>
</table>

https://en.wikipedia.org/wiki/Anscombe%27s_quartet
Why visualize data?

- Four groups
- 11 observations \((x, y)\) per group

https://en.wikipedia.org/wiki/Anscombe%27s_quartet
R base graphics

- `plot()` generic x-y plotting
- `barplot()` bar plots
- `boxplot()` box-and-whisker plot
- `hist()` histograms

Don’t use barplots

https://cogtales.wordpress.com/2016/06/06/congratulations-barbarplots/
R base graphics

- `stats::heatmap()` - basic heatmap

Alternatives:

- `gplots::heatmap.2()` - an extension of heatmap
- `heatmap3::heatmap3()` - another extension of heatmap
- `ComplexHeatmap::Heatmap()` - highly customizable, interactive heatmap

Other options:

- `pheatmap::pheatmap()` - grid-based heatmap
- `NMF::aheatmap()` - another grid-based heatmap
More heatmaps

- `fheatmap::fheatmap()` - heatmap with some ggplot2
- `gapmap::gapmap()` - gapped heatmap (ggplot2/grid)

Interactive heatmaps:

- `d3heatmap::d3heatmap()` - interactive heatmap in d3
- `heatmaply::heatmaply()` - interactive heatmap with better dendrograms

Compare clusters

- `dendextend` package - make better dendrograms, compare them with ease

Other useful plots

- `qqnorm()`, `qqline()`, `qqplot()` - distribution comparison plots
- `pairs()` - pair-wise plot of multivariate data

TOC-Some-Great-R-Functions
Special plots

- `vioplot()`: Violin plot, [https://cran.r-project.org/web/packages/vioplot/](https://cran.r-project.org/web/packages/vioplot/)
- `PiratePlot()`: violin plot enhanced.
  `install_github("ndphillips/yarrrr"), [http://nathanieldphillips.com/](http://nathanieldphillips.com/)
- `beeswarm()`: The Bee Swarm Plot, an Alternative to Stripchart, [https://cran.r-project.org/web/packages/beeswarm/index.html](https://cran.r-project.org/web/packages/beeswarm/index.html)
Saving plots

- Save to PDF

```r
pdf("filename.pdf", width = 7, height = 5)
plot(1:10, 1:10)
dev.off()
```

- Other formats: `bmp()`, `jpg()`, `pdf()`, `png()`, or `tiff()

- Learn more `?Devices`
R base graphic cheat-sheet

Data manipulation
80% of your work will be data preparation

- getting data (from databases, spreadsheets, flat-files)
- performing exploratory/diagnostic data analysis
- reshaping data
- visualizing data

80% of your work will be data preparation

- Filtering rows (to create a subset)
- Selecting columns of data (i.e., selecting variables)
- Adding new variables
- Sorting
- Aggregating
- Joining

Dplyr: A grammar of data manipulation

https://github.com/hadley/dplyr

install.packages("dplyr")
The pipe `%>%` operator

- Pipe output of one command into an input of another command - chain commands together
- Think about the “|” operator in Linux
- Read as “then”. Take the dataset, then do …

```r
library(dplyr)
library(ggplot2)
data(diamonds)
head(diamonds)
diamonds %>% head
summary(diamonds$price)
diamonds$price %>% summary(object = .)
```
**dplyr::filter()**

- Filter (select) rows based on the condition of a column

```r
diamonds %>% head
def.diamonds_ideal <- filter(diamonds, cut == "Ideal")
def.diamonds_ideal <- diamonds %>% filter(cut == "Ideal")
```
**dplyr::select()**

- Select columns from the dataset by names

```r
df.diamonds_ideal %>% head
select(df.diamonds_ideal, carat, cut, color, price, clarity)
```

```r
df.diamonds_ideal <- df.diamonds_ideal %>% select(., carat, cut,
```
**dplyr::mutate()**

- Add columns to your dataset

```r
df.diamonds_ideal %>% head
mutate(df.diamonds_ideal, price_per_carat = price/carat)
df.diamonds_ideal <- df.diamonds_ideal %>% mutate(price_per_carat = price/carat)
```
Sort your data by columns

df.diamonds_ideal %>% head
arrange(df.diamonds_ideal, price)
df.diamonds_ideal %>% arrange(price, price_per_carat)
dplyr::summarize()

- Summarize columns by custom summary statistics

```r
summarize(df.diamonds_ideal, length = n(), avg_price = mean(price))
df.diamonds_ideal %>% summarize(length = n(), avg_price = mean(price))
```
**dplyr::group_by()**

- Summarize *subsets of* columns by custom summary statistics

```r
group_by(diamonds, cut) %>% summarize(mean(price))
group_by(diamonds, cut, color) %>% summarize(mean(price))
```
The power of pipe \( \%\%\) 

- Summarize *subsets of columns* by custom summary statistics

```r
diamonds %>%
  filter(tbl_df(diamonds), cut == "Ideal") %>%
  arrange(price) %>%
  mutate(price_per_carat = price/carat) %>%
  arrange(price_per_carat)
```
ggplot2 - the grammar of graphics
ggplot2 package

http://ggplot2.org/

install.packages("ggplot2")

**ggplot2**

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

**Documentation**

ggplot2 documentation is now available at docs.ggplot2.org.
## The basics of ggplot2 graphics

- Data mapped to graphical elements
- Add graphical layers and transformations
- Commands are chained with “+” sign

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>The raw data that you want to plot</td>
</tr>
<tr>
<td>Aesthetics aes()</td>
<td>How to map your data on x, y axis, color, size, shape (aesthetics)</td>
</tr>
<tr>
<td>Geometries geom_</td>
<td>The geometric shapes that will represent the data</td>
</tr>
</tbody>
</table>

data +
aesthetic mappings of data to plot coordinates +
geometry to represent the data
Examples of ggplot2 graphics

```r
diamonds %>% filter(cut == "Good", color == "E") %>%
  ggplot(aes(x = price, y = carat)) +
  geom_point()  # aes(size = price) +
```

Try other geoms

```r
  geom_smooth()  # method = lm
  geom_line()
  geom_boxplot()
  geom_bar(stat="identity")
  geom_histogram()
```
## Fine tuning ggplot2 graphics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facets</td>
<td>facet_{Split one plot into multiple plots based on a grouping variable}</td>
</tr>
<tr>
<td>Scales</td>
<td>scale_{Maps between the data ranges and the dimensions of the plot}</td>
</tr>
<tr>
<td>Visual Themes</td>
<td>theme_{The overall visual defaults of a plot: background, grids, axe, default typeface, sizes, colors, etc.}</td>
</tr>
<tr>
<td>Statistical</td>
<td>stat_{Statistical summaries of the data that can be plotted, such as quantiles, fitted curves (loess, linear models, etc.), sums etc.}</td>
</tr>
<tr>
<td>Coordinate systems</td>
<td>coord_{Expressing coordinates in a system other than Cartesian}</td>
</tr>
</tbody>
</table>
Putting it all together

diamonds %>%  
  filter(cut == "Ideal") %>%  
  ggplot(aes(price)) +  
  geom_histogram() +  
  facet_wrap(~ color) +  
  ggtitle("Diamond price distribution per color") +  
  labs(x="Price", y="Count") +  
  theme(panel.background = element_rect(fill="lightblue")) +  
  theme(plot.title = element_text(family="Trebuchet MS", size=)) +  
  theme(axis.title.y = element_text(angle=0)) +  
  theme(panel.grid.minor = element_blank())
Other resources

- **GoogleVis** for R, https://cran.r-project.org/web/packages/googleVis/vignettes/googleVis_examples.html
- **ggbio** - grammar of graphics for genomic data, http://www.tengfei.name/ggbio/