Advanced Plotting with ggplot2

Algorithm Design & Software Engineering

November 13, 2016

Stefan Feuerriegel

Today's Lecture

Objectives

1. Distinguishing different types of plots and their purpose
2. Learning the grammar of graphics
3. Create high-quality plots with ggplot2

Outline

Outline

Motivation

Why plotting?

- Visualizations makes it easier to understand and explore data
- Common types of plots: bar chart, histogram, line plot, scatter plot, box plot, pirate plot, ...

Plotting with ggplot2 in R

- Built-in routines cover most types, yet the have no consistent interface and limited flexibility
- Package ggplot2 is a powerful alternative
  - Abstract language that is flexible, simple and user-friendly
  - Nice aesthetics by default
  - Themes for common look-and-feel
- "gg" stands for "grammar of graphics"
- Limited to 2D plots (3D plots not supported)
- Commonly used by New York Times, Economics, ...

Example with ggplot2

- Load package

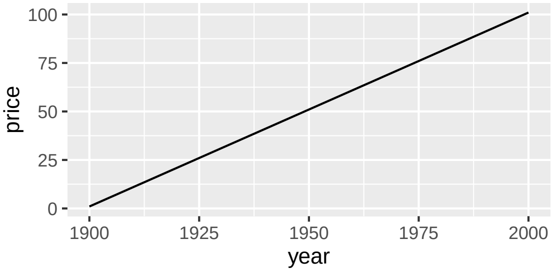
```r
library(ggplot2)
```

- Create sample data

```r
line_data <- data.frame(year=1900:2000, price=1:101)
```

- Visualize data frame as line plot

```r
ggplot(line_data, aes(x=year, y=price)) +
  geom_line()
```



Calls to ggplot2

General format

```
ggplot(data, aes(x=variable_x, y=variable_y)) +
  geom_*() +
  additional_modifications()
```

- `ggplot()` expects a data frame (not: matrix) as a first input, followed by the aesthetics that map variables by name onto axes
- Building blocks are concatenated via +
- $*$ is any of the supported plot types
- The `geom_*()` can overwrite previous aesthetics

  -
    ```
    ggplot(data) +
      geom_line(aes(x=variable_x, y=variable_y1)) +
      geom_line(aes(x=variable_x, y=variable_y2))
    ```

  -
    ```
    ggplot(data, aes(x=variable_x)) +
      geom_line(aes(y=variable_y1)) +
      geom_line(aes(y=variable_y2))
    ```

Terminology

- **Data**: underlying information to be visualized
- **Aesthetics**: controls the color/shape/... of observations and which variables go on the x- and y-axis
- **Geometry**: geometric objects in the plot; e. g. points, lines, bars, polygons, ...
- **Layers**: individual plots, i. e. calls to `geom_*()`
- **Facets**: creates panels of sub-plots
- **Scales**: sets look-and-feel of axes
- **Themes**: overall color palette and layout of plot
- **Statistics**: transformations of the data before display
- **Legends**: appearance and position of legend
  - Each layer consists of data and aesthetics, plus additional customizations
  - A plot can have a one or an arbitrary number of layers

Aesthetics

- Aesthetics `aes(...)` set “what you see”
  - Variables which go on x- and y-axis
  - Color of outer border
  - Fill color of inside area
  - Shape of points
  - Line type
  - Size of points and lines
  - Grouping of values
- Expect a column name representing the variable
- Short form by `aes(x, y)` where identifiers `x=` and `y=` are omitted

Wide vs. Long Data

Data format

- **Wide** data: multiple measurements for the same subject, each in a different column
- **Long** data: subjects have multiple rows, each with one measurement

Example

Wide format

Company	**Sales Drinks**	**Sales Food**
A	300	400
B	200	100
C	50	0

$\Rightarrow$

Long format

Company	**Category**	**Sales**
A	Drinks	300
A	Food	400
B	Drinks	200
B	Food	100
C	Drinks	50
C	Food	0

Note: ggplot2 requires data in long format

Conversion Between Long and Wide Data

- Prepare sample data

```r
d_wide <- data.frame(Company = c("A", "B", "C"),
                     SalesDrinks = c(300, 200, 50),
                     SalesFood = c(400, 100, 0))
```

- Load necessary package `reshape2`

```r
library(reshape2)
```

- Call function `melt(data_wide, id.vars=v)` to convert wide data into a long format where `v` identifies the subject

```r
melt(d_wide, id.vars="Company")

##   Company    variable value
## 1       A SalesDrinks   300
## 2       B SalesDrinks   200
## 3       C SalesDrinks    50
## 4       A   SalesFood   400
## 5       B   SalesFood   100
## 6       C   SalesFood     0
```

Outline

Plot Types

- ggplot2 ships the following geometric objects (geoms) amongst others
- Function names start with `geom_*()`

Two variables

- Scatter plot (also named point plot) `geom_point()`
- Line plot `geom_line()`
- Area chart `geom_area()`

One variable (discrete)

- Bar chart `geom_bar()`

One variable (continuous)

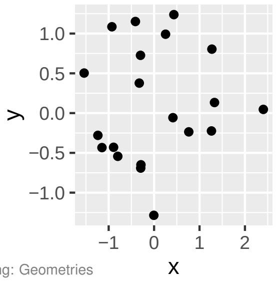
- Histogram `geom_histogram()`
- Boxplot `geom_boxplot()`
- Density plot `geom_density()`

Scatter Plot

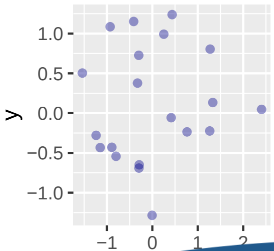
- A scatter plot displays each observation as a geometric point
- Optional arguments: `alpha` (transparency), `size`, `color`, `shape`

```r
points <- data.frame(x=rnorm(20), y=rnorm(20))
p1 <- ggplot(points, aes(x, y)) +
  geom_point()
p2 <- ggplot(points, aes(x, y)) +
  geom_point(alpha=0.4, color="darkblue")
```

```
p1
```

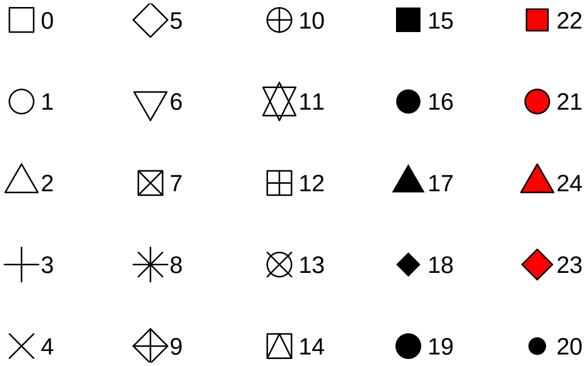


```
p2
```



Point Shapes

- Argument `shape` accepts different values

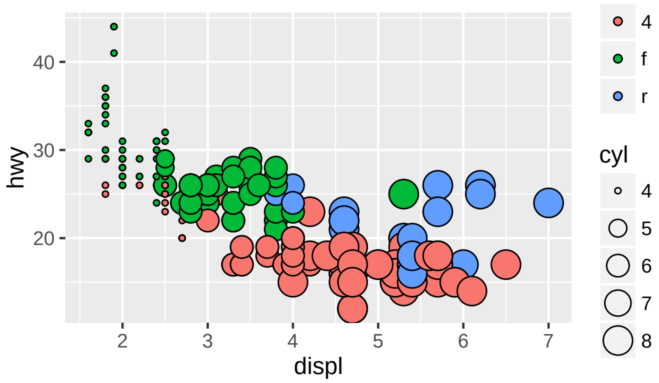


- Shapes 21–24 distinguish two colors:
  - A border color (argument: `color`)
  - A fill color (argument: `fill`)

Scatter Plot

- Aesthetics can also change size, shape or color based on variables

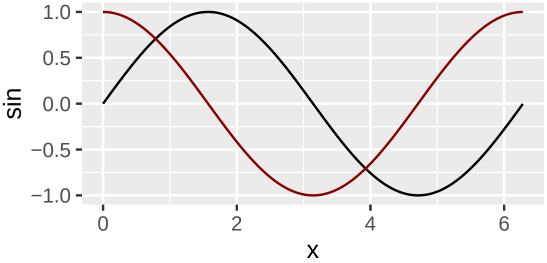
```
ggplot(mpg, aes(x=displ, y=hwy)) +
  geom_point(aes(size=cyl, fill=drv), shape=21)
```



Line Plot

- Line plot displays points as a connected line

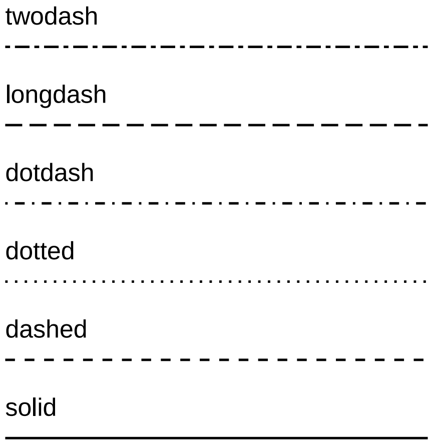
```
x <- seq(0, 2*pi, by=0.01)
data_sin_cos <- data.frame(x=x, sin=sin(x), cos=cos(x))

ggplot(data_sin_cos, aes(x)) +
  geom_line(aes(y=sin)) +
  geom_line(aes(y=cos), color="darkred")
```



- Optional arguments: `color`, `linetype`, `size`, `group`

Line Types

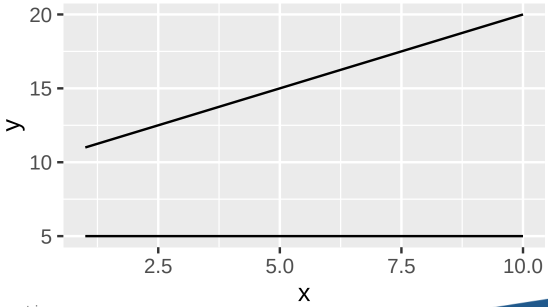
- Argument `linetype` picks a line type based the following identifiers



Line Plot

- Long data allows for efficient grouping and simpler plots
- Argument `group` denotes the variable with the group membership
- Alternative is to use `color` for different colors

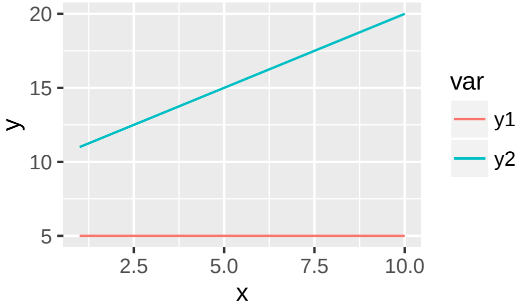
```
data_lines2 <- data.frame(x=c(1:10, 1:10),
                          var=c(rep("y1", 10), rep("y2", 10))
                          y=c(rep(5, 10), 11:20))
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, group=var))
```



Line Plot

- Grouping can occur through all aesthetics
- Common is to use `color` for different colors

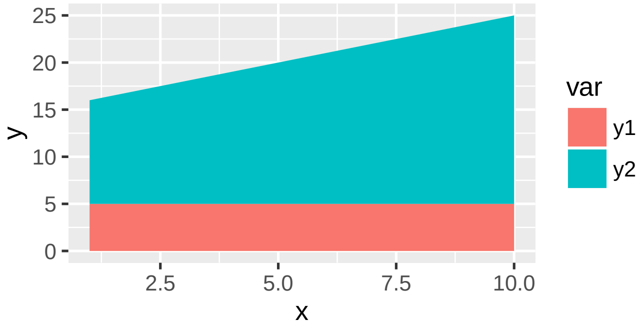
```
data_lines2 <- data.frame(x=c(1:10, 1:10),
                          var=c(rep("y1", 10), rep("y2", 10))
                          y=c(rep(5, 10), 11:20))
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var))
```



Area Chart

- Similar to a line plot, but the area is filled in color
- Individual areas are mapped via `group` and colored via `fill`
- `position="stack"` stacks the areas on top of each other

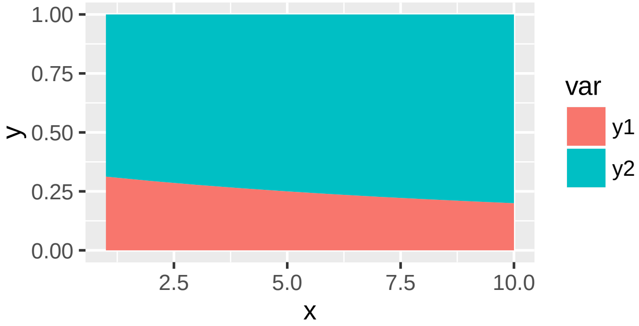
```
ggplot(data_lines2) +
  geom_area(aes(x=x, y=y, fill=var, group=var),
            position="stack")
```



Area Chart

- Argument `position="fill"` shows relative values for each group out of 100 %

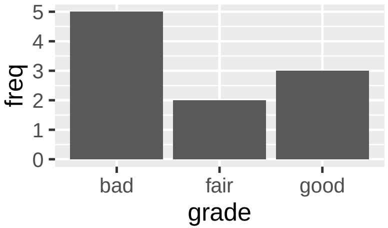
```
ggplot(data_lines2) +
  geom_area(aes(x=x, y=y, fill=var, group=var),
            position="fill")
```



Bar Chart

- Bar chart compares values, counts and statistics among categories
- The x-axis usually displays the discrete categories
- The y-axis depicts the given value (`stat="identity"`) or also transformed statistics

```
grades_freq <- data.frame(grade=c("good", "fair", "bad"),
                          freq=c(3, 2, 5))
ggplot(grades_freq) +
  geom_bar(aes(grade, freq), stat="identity")
```

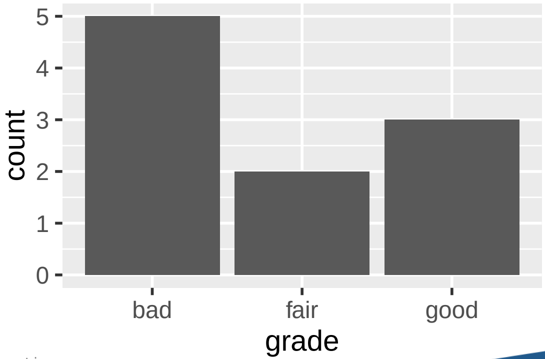


- Categories are sorted alphabetically by default

Bar Chart

- `stat="count"` automatically counts the frequency of observations

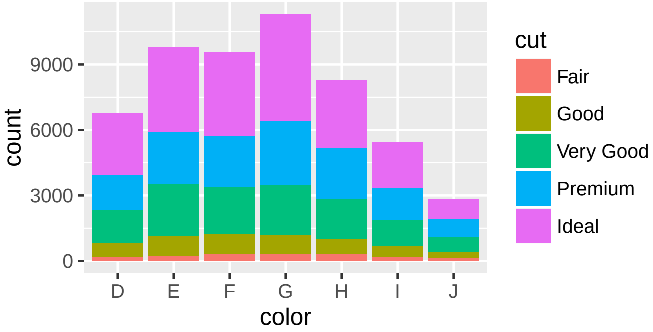
```
grades <- data.frame(grade=c("good", "good", "good",
                             "fair", "fair",
                             "bad", "bad", "bad",
                             "bad", "bad"))
ggplot(grades) +
  geom_bar(aes(x=grade), stat="count")
```



Stacked Bar Chart

- Group membership controlled by fill color

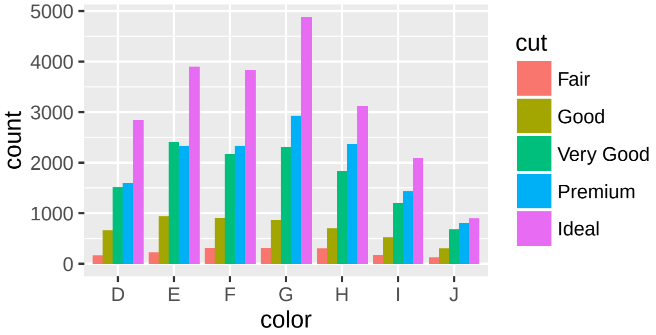
```
ggplot(diamonds) +
  geom_bar(aes(x=color, fill=cut), stat="count")
```



Grouped Bar Chart

- Bars are displayed next to each other via `position="dodge"`

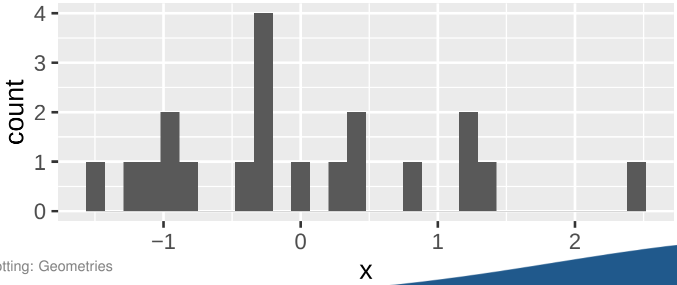
```
ggplot(diamonds) +
  geom_bar(aes(x=color, fill=cut), stat="count",
           position="dodge")
```



Histogram

- Histogram shows frequency of continuous data by dividing the range of values into bins
- Each bar then denotes the frequency of data falling into that bin
- Illustrates the distribution of the data

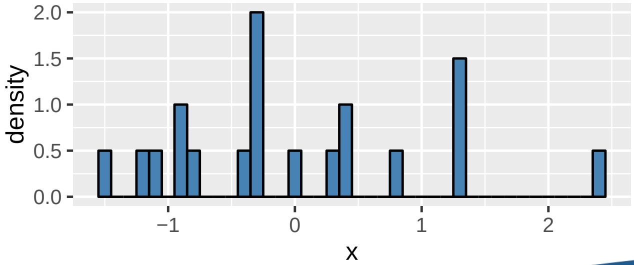
```
ggplot(points) +
  geom_histogram(aes(x))

## `stat_bin()` using `bins = 30`.  Pick better value with
`binwidth`.
```



Histogram

- Optional arguments: border color (`color`), fill color (`fill`), width of the bins (`binwidth`)
- ggplot automatically defines new variables (`..count..` and `..density..`) that can be used in the aesthetics
- `y=..density..` displays density on y-axis instead of frequency

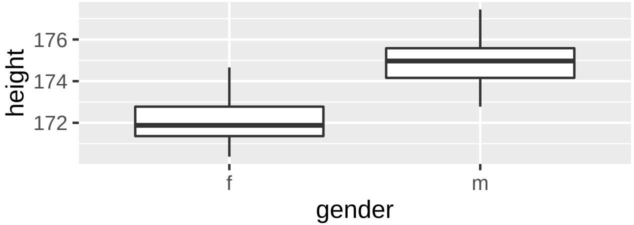
```r
ggplot(points) +
  geom_histogram(aes(x, y=..density..), binwidth=0.1,
                           fill="steelblue", colour="black")
```



Box Plot

- Box plots visualize distribution by highlighting median and quartiles

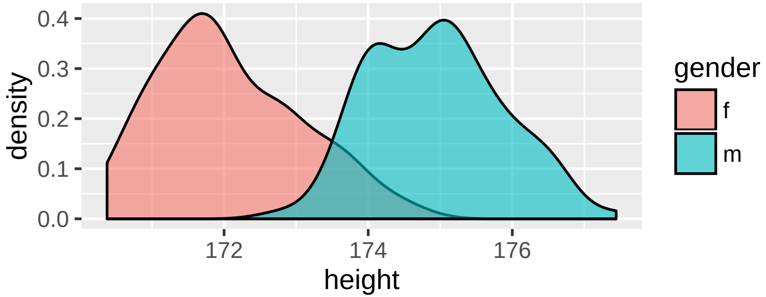
```
height <- data.frame(gender=c(rep("m", 100), rep("f", 100)),
                     height=c(rnorm(100, mean=175),
                              rnorm(100, mean=172)))
ggplot(height) +
  geom_boxplot(aes(gender, height))
```



Density Plot

- Estimates the density as a mean to approximate the distribution
- Smooth alternative of a histogram
- Optional argument: `alpha` allows colors to be transparent

```
ggplot(height) +
  geom_density(aes(x=height, fill=gender),
               stat="density", alpha=0.6)
```



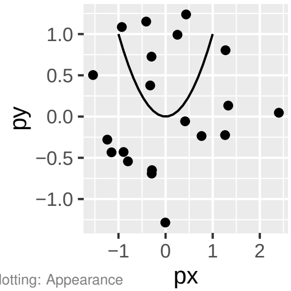
Outline

Outline

Multiple Layers

- Concatenation allows for combining several layers
- Each layer has it own aesthetics

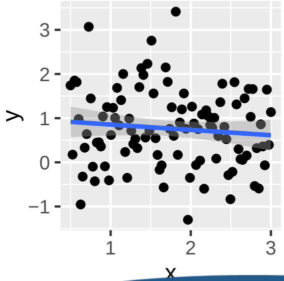
```
df <- data.frame(px=rnorm(20), py=rnorm(20),
                 lx=seq(-1, +1, length.out=20))
df$ly <- df$lx^2

ggplot(df) +
  geom_point(aes(px, py)) +
  geom_line(aes(lx, ly))
```



Smoothing Layers

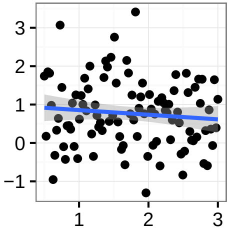
- Smoothing layer `geom_smooth` implements trend curves
  - Linear trend (`method="lm"`)
  - Local polynomial regression (`method="loess"`) with smoothing parameter `span`
  - Generalized additive model (`method="gam"`)
- Variable choice is also controlled by aesthetics `aes(x, y)`
- Gray shade highlights the 95 % confidence interval

```r
df <- data.frame(x=seq(0.5, 3,
                        length.out=100))
df$y <- sin(df$x) + rnorm(100)

ggplot(df, aes(x, y)) +
  geom_point() +
  geom_smooth(method="lm")
```

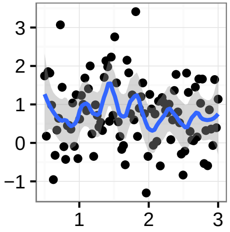


Smoothing Layers

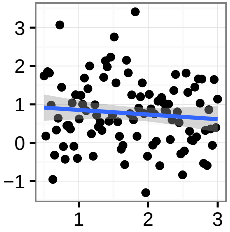
`method="lm"`



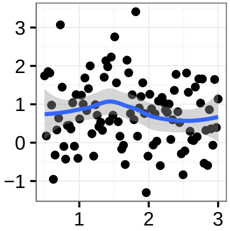
`method="loess", span=0.25`



`method="gam"`



`method="loess", span=0.75`



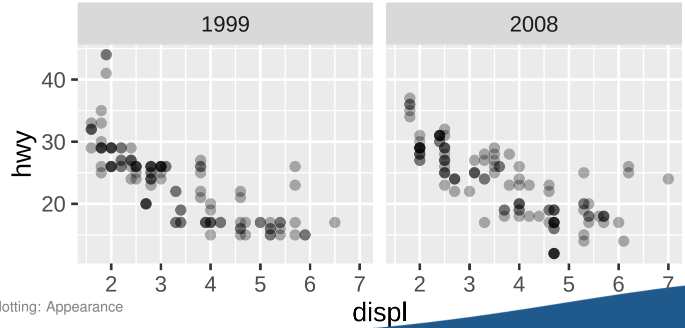
Outline

Facets

- Facets display a grid of plots stemming from the same data
- Command: `facet_grid(y ~ x)` specifies grouping variables
- By default, the same axis resolution is used on adjacent plots

Example with 1 group on x-axis

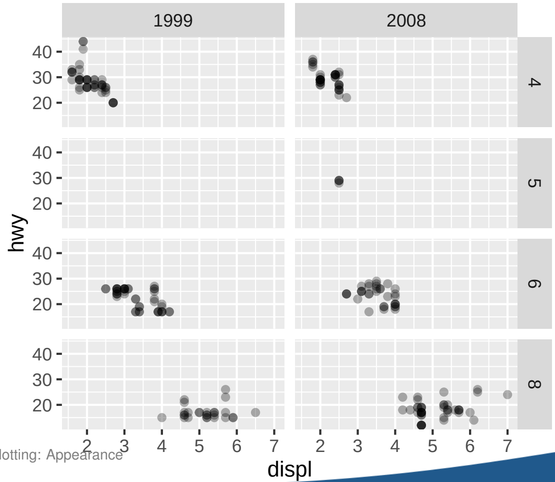
```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(alpha = 0.3) +
  facet_grid(. ~ year)
```



Facets

Example with 2 groups on x- and y-axis

```
ggplot(mpg, aes(displ, hwy)) +
  geom_point(alpha = 0.3) +
  facet_grid(cyl ~ year)
```



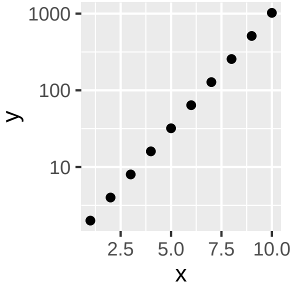
Outline

Scales

Scales control the look of axes, especially for continuous and discrete data

- `scale_<axis>_log10()` uses log-scale on axis

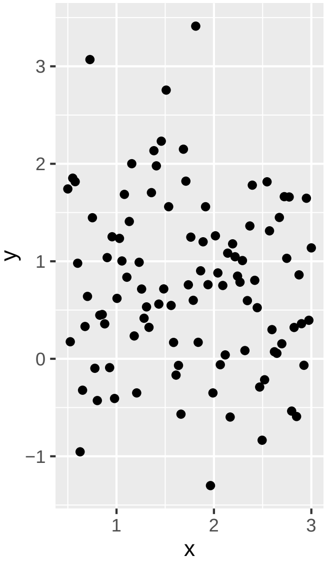
```
exp_growth <- data.frame(x=1:10,
                         y=2^(1:10))
ggplot(exp_growth, aes(x, y)) +
  geom_point() +
  scale_y_log10()
```



Scales

- `coord_equal()` enforces an equidistant scaling on both axes

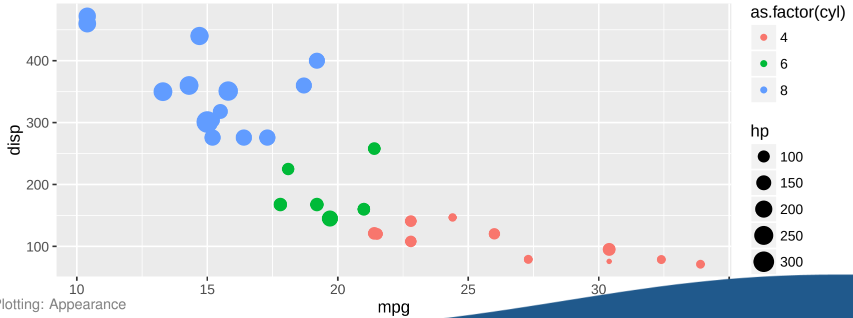
```
ggplot(df, aes(x, y)) +
  geom_point() +
  coord_equal()
```



Geometry Layout

- Changes to geometry layout links to the use of aesthetics
- Additional function call to `scale_<aestetics>_<type>(...)`
  1. Aesthetic to change, e. g. `color`, `fill`, `linetype`, ...
  2. Variable type controls appearance, e. g. `gradient` (continuous scale), `hue` (discrete values), `manual` (manual breaks), ...

```r
ggplot(mtcars, aes(x=mpg, y=disp)) +
  geom_point(aes(size=hp, color=as.factor(cyl)))
```

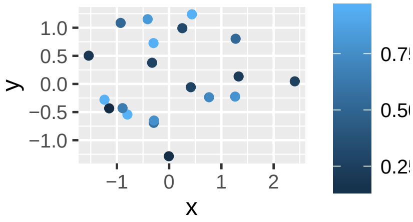


scale_color_gradient

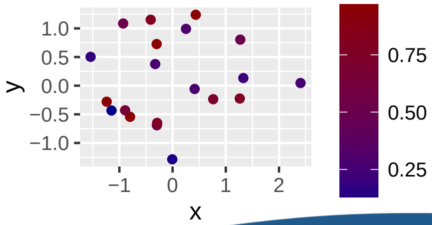
- Color gradient stems from a range between two colors
  $\rightarrow$ Arguments: `low`, `high`
- Useful for visualizing continuous values

```
points_continuous <- cbind(points, z=runif(20))
p <- ggplot(points_continuous) +
  geom_point(aes(x=x, y=y, color=z))
```

```
p + scale_color_gradient()
```



```
p + scale_color_gradient(low="darkblue",
                         high="darkred")
```

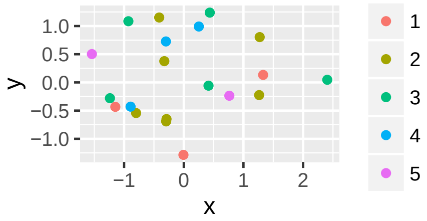


scale_color_hue

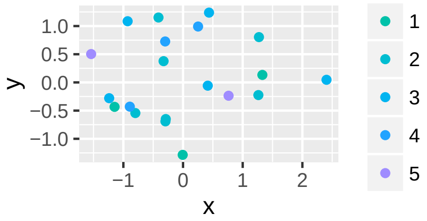
- Uses disjunct buckets of colors for visualizing discrete values
- Requires source variable to be a factor

```r
points_discrete <- cbind(points,
                         z=as.factor(sample(5, 20, replace=TRUE)))
p <- ggplot(points_discrete) +
  geom_point(aes(x=x, y=y, color=z))
```

```r
p + scale_color_hue()
```



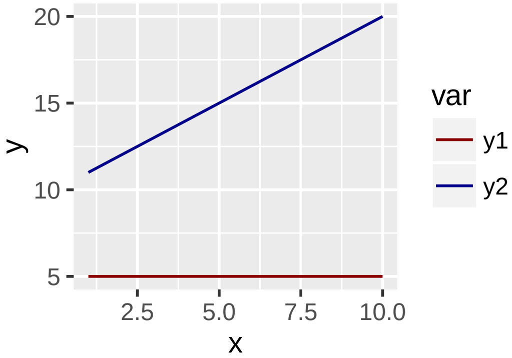
```r
p + scale_color_hue(h=c(180, 270))
```



scale_color_manual

- Specifies colors for different groups manually
- Argument `values` specifies a vector of new color names

```
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var)) +
  scale_color_manual(values=c("darkred", "darkblue"))
```



Color Palettes

- Built-in color palettes change color scheme
- Distinguished by discrete and continuous source variables
  1. **Discrete** values and colors via `scale_color_brewer()`
  2. **Continuous** values and colors via `scale_color_distiller()`
- Further customizations
  - Overview of color palettes:
    `http://www.cookbook-r.com/Graphs/Colors_(ggplot2)`
  - Package `ggtheme` has several built-in schemes:
    `https://cran.r-project.org/web/packages/ggthemes/vignettes/ggthemes.html`
  - Color picker:
    `http://www.colorbrewer2.org/`

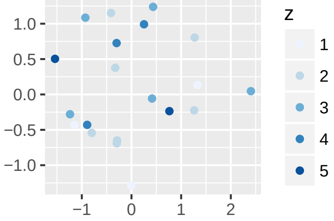
Discrete Color Palettes

- `scale_color_brewer` accesses built-in color palettes for discrete values

```
pd <- ggplot(points_discrete) +
  labs(x="", y="") +
  geom_point(aes(x, y, color=z))
```

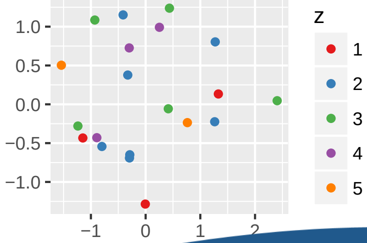
Default

```
pd + scale_color_brewer()
```



Intense colors

```
pd + scale_color_brewer(palette="Set1")
```



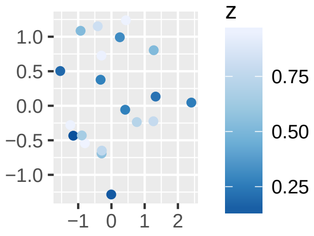
Continuous Color Palettes

- `scale_color_distiller` accesses built-in color palettes for continuous values

```
pc <- ggplot(points_continuous) +
  labs(x="", y="") +
  geom_point(aes(x, y, color=z))
```

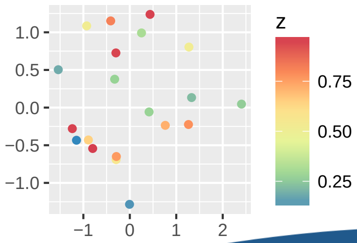
Default

```
pc + scale_color_distiller()
```



Spectral colors

```
pc + scale_color_distiller(palette="Spectral")
```

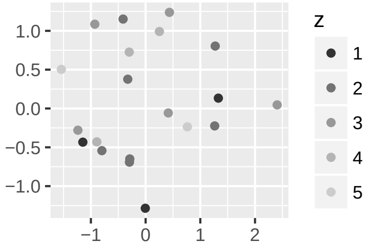


Gray-Scale Coloring

- No unique identifier for gray-scale coloring
  1. `scale_color_gray()` colors discrete values in gray-scale
     → Attention: "`grey`" as used in British English
  2. `scale_color_gradient()` refers to a continuous spectrum

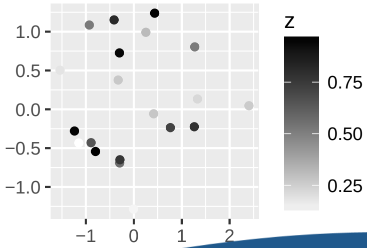
Discrete values

```
pd + scale_color_grey()
```



Continuous values

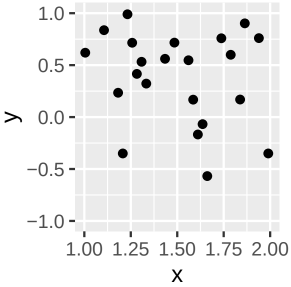
```
pc + scale_color_gradient(low="white",
                          high="black")
```



Ranges

- Crop plot to ranges via `xlim(range)` or `ylim(range)`

```
ggplot(df, aes(x, y)) +
  geom_point() +
  xlim(c(1, 2)) +
  ylim(c(-1, +1))
```

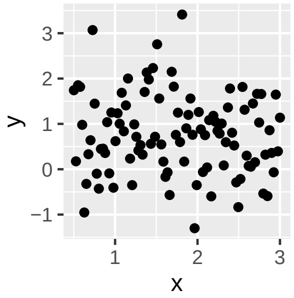


Outline

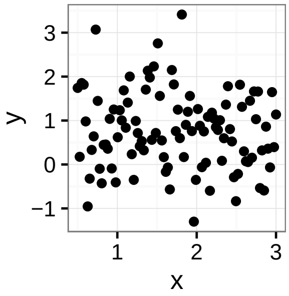
Themes

- Themes further customize the appearance of plots
- Printer-friendly theme `theme_bw()` for replacing the gray background

```r
ggplot(df, aes(x, y)) +
  geom_point()
```



```r
ggplot(df, aes(x, y)) +
  geom_point() +
  theme_bw()
```



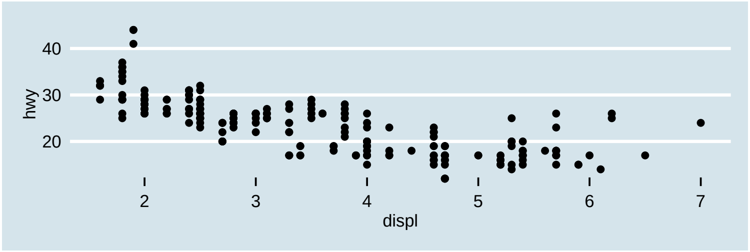
Themes

- Package `ggthemes` provides further styles

```r
library(ggthemes)
```

Example with the style from *The Economist*

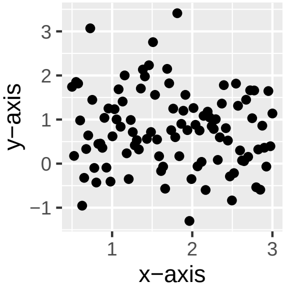
```r
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  theme_economist()
```



Labels

- Change labels via `labs(...)`

```
ggplot(df, aes(x, y)) +
  geom_point() +
  labs(x = "x-axis", y = "y-axis")
```



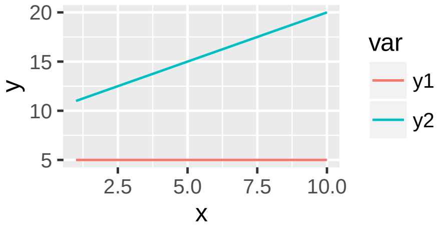
Recommendation: don't use titles in plots
$\rightarrow$ Instead of titles, better place details in the caption of scientific papers

Outline

Legend

- Legends are placed automatically for each aesthetic in used
- Examples: `group`, `color`, ...

```
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var))
```



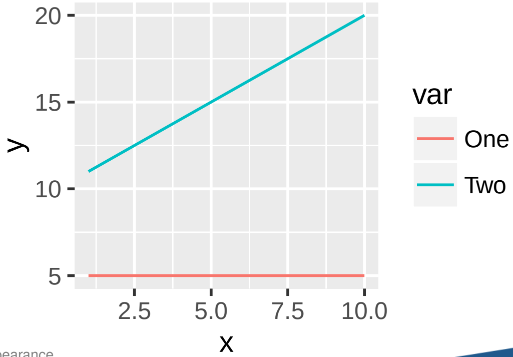
- Frequent changes include
  1. Data is in long format and should be renamed
  2. Data is in long format and should be customized
  3. Data is in wide format and each `geom_*` should be customized

Legend

Case 1: Data is in long format and should be renamed

- Add `scale_<aesthetics>_discrete(...)` to overwrite matching
- Argument `labels` specifies new labels

```
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var)) +
  scale_color_discrete(labels=c("One", "Two"))
```

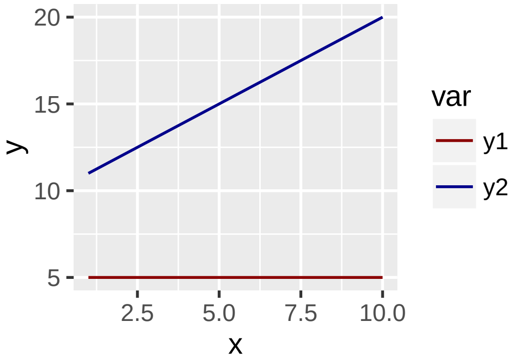


Legend

Case 2: Data is in long format and should be customized

- Add `scale_<aesthetics>_manual` to change appearance
- Argument `values` specifies new attributes (e. g. color)

```
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var)) +
  scale_color_manual(values=c("darkred", "darkblue"))
```

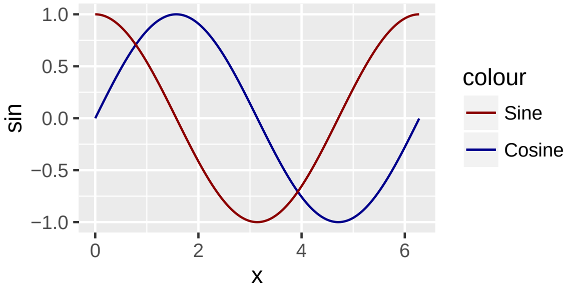


Legend

Case 3: Data is in wide format and each `geom_*` should be customized

- Add additional aesthetics with string identifier
- Change appearance with `scale_<aesthetics>_manual()`

```
ggplot(data_sin_cos, aes(x)) +
  geom_line(aes(y=sin, color="sin")) +
  geom_line(aes(y=cos, color="cos")) +
  scale_color_manual(labels=c("Sine", "Cosine"),
                     values=c("darkred", "darkblue"))
```

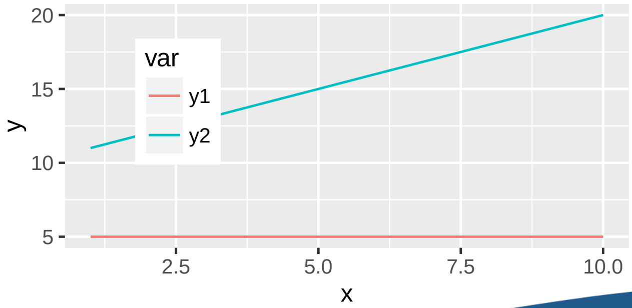


- **Recommendation:** better convert to long fo

Legend Position

- Default position of legend is outside of plot
- `theme(legend.position="none")` hides the legend
- `theme(legend.position=c(x, y))` moves it inside the grid
- $x, y \in [0, 1]$ are relative positions starting from the bottom-left corner

```r
ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var)) +
  theme(legend.position=c(0.2, 0.6))
```



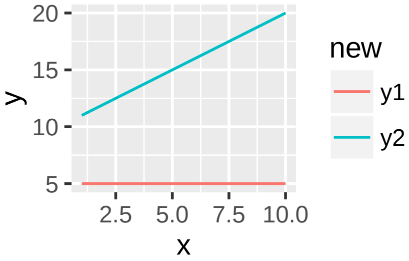
Legend Title

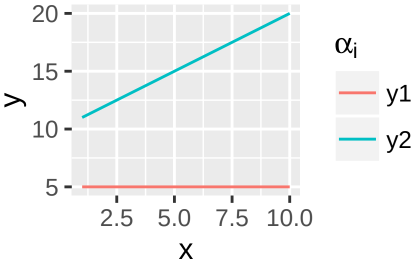
- Legend title is set inside `scale_<aesthetics>_<type>(...)`
- Passed as the first argument or argument `name`
- Displays maths via `expression(...)`

```
p <- ggplot(data_lines2) +
  geom_line(aes(x=x, y=y, color=var))
```

```
p + scale_color_discrete(name="new")
```

```
p + scale_color_discrete(expression(alpha[i]))
```





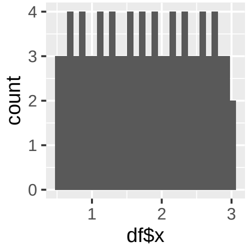
Outline

qplot

- `qplot(x, y)` is a wrapper similar to `plot(...)`

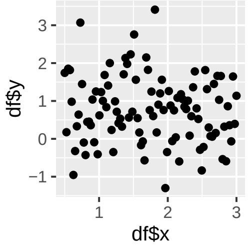
Histogram

```r
qplot(df$x)
```



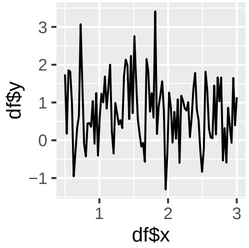
Point plot

```r
qplot(df$x, df$y)
```



Line plot

```r
qplot(df$x, df$y,
      geom="line")
```

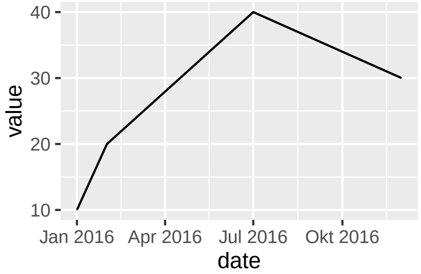


Date and Time

- Values of type date or time are formatted automatically

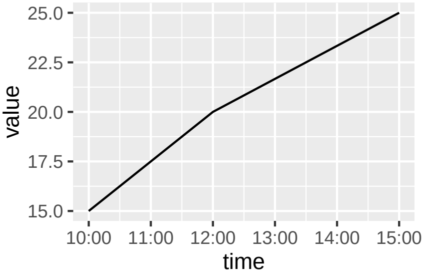
Date

```r
dates <- as.Date(c("2016-01-01", "2016-02-01",
                   "2016-07-01", "2016-12-01"))
sales <- data.frame(date=dates,
                    value=c(10, 20, 40, 30))
ggplot(sales, aes(date, value)) +
  geom_line()
```



Time

```r
times <- as.POSIXct(c("2001-01-01 10:00",
                      "2001-01-01 12:00",
                      "2001-01-01 15:00"))
temp <- data.frame(time=times,
                   value=c(15, 20, 25))
ggplot(temp, aes(time, value)) +
  geom_line()
```



Maps

- Package `ggmap` allows to plot geometries on a map

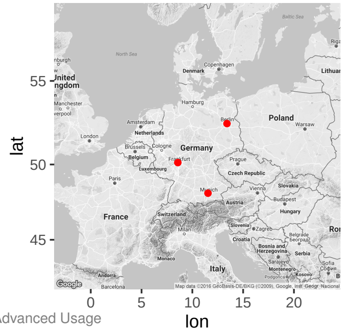
```r
library(ggmap)
```

- Download map with `get_map(...)`

```r
map <- get_map("Germany", zoom=5, color="bw")
```

- Coordinates are given as longitude/latitude

```r
geo <- data.frame(lat=c(52.52, 50.12, 48.15),
                  lon=c(13.41, 8.57, 11.54))
ggmap(map) +
  geom_point(data=geo, aes(lon, lat), color="red")
```



lat
55
50
45
lon
0
5
10
15
20
Germany
Poland
France
Italy
Berlin
Frankfurt
Munich
Map data ©2016 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional

Exporting Plots

- Workflow is greatly accelerated when exporting plots automatically
- PDF output is preferred in LaTeX, PNG for Word
- `ggsave(filename)` exports the last plot to the disk
  1. Export as PNG
     ```
     ggsave("plot.png")
     ```
  2. Export as PDF
     ```
     ggsave("plot.pdf")
     ```
- File extension specifies format implicitly
- Alternative arguments specify filename and size (i. e. resolution)

```
p <- ggplot(df, aes(x, y))
ggsave(p, file="/path/plot.pdf",
       width=6, height=4)
```

Outline

Further Reading

Online resources

- Official ggplot2 documentation
  `http://docs.ggplot2.org/current/`
  → Collection of reference materials and examples how parameters affect the layout
- Cookbook for R Graphs
  `http://www.cookbook-r.com/Graphs/`
  → Collection of problem-solution pairs by plot type with different layout customizations
- Introduction to R Graphics with ggplot2
  `http://en.slideshare.net/izahn/rgraphics-12040991`
  → Introductory presentation with many examples
- ggplot2 Essentials
  `http://www.sthda.com/english/wiki/ggplot2-essentials`
  → Overview of different plots and available options for customization

Books

- Wickham (2016). “ggplot2: Elegant Graphics for Data Analysis”, 2nd ed., Springer.