Introduction to ggplot2

Anne Segonds-Pichon, Simon Andrews
v. 2019.4-2
General structure

- A graph is made up of layers.
- `geom()` visual elements = geometric objects: points, bars ...
- `aes()` aesthetics properties (colour, shape, size ...)
  - control appearance and locations of `geom()`
General Structure

• `ggPlot` objects are built up in a variable created by the `ggplot` function.

• The plot is drawn when the variable is printed to the console.

• Multiple layers are added by using the ‘+’ operator. They are added to the variable rather than being drawn on the plot.
General Structure

- Call the `ggplot` function and specify the aesthetics you want to use
  - You need to specify enough for the type of geometry you want to use

- Add a `geom_` function to specify what representation to use.

- [optional] Add further `geom_` calls to add more data or other annotation views

- Print the result
geoms() and aes()

- **aes()**: what are we going to plot. Which parts of our data are we plotting, and which of the aspects of our plot are they going to modify

- **geom()**: geometric object – what type of plot or annotation are we going to use to display our chosen aesthetics

<table>
<thead>
<tr>
<th>Geoms</th>
<th>Required</th>
<th>optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom_bar()</td>
<td>x: the variable to plot on the x-axis</td>
<td>colour, size, fill, linetype, alpha (transparency)</td>
</tr>
<tr>
<td>geom_point()</td>
<td>x: the variable to plot on the x-axis</td>
<td>shape, colour, size, fill, alpha</td>
</tr>
<tr>
<td></td>
<td>y: the variable to plot on the y-axis</td>
<td></td>
</tr>
<tr>
<td>geom_line()</td>
<td>x: the variable to plot on the x-axis</td>
<td>colour, size, linetype, alpha</td>
</tr>
<tr>
<td></td>
<td>y: the variable to plot on the y-axis</td>
<td></td>
</tr>
<tr>
<td>geom_smooth()</td>
<td>x: the variable to plot on the x-axis</td>
<td>colour, size, fill, linetype, alpha</td>
</tr>
<tr>
<td></td>
<td>y: the variable to plot on the y-axis</td>
<td></td>
</tr>
<tr>
<td>geom_histogram()</td>
<td>x: the variable to plot on the x-axis</td>
<td>colour, size, fill, linetype, alpha</td>
</tr>
<tr>
<td>geom_boxplot()</td>
<td>x: the variable to plot on the x-axis</td>
<td>colour, size, fill, alpha</td>
</tr>
<tr>
<td>geom_text()</td>
<td>x: the horizontal coordinate of where the text should be placed</td>
<td>colour, size, angle, hjust(horizontal adjustment), vjust (vertical adjustment), alpha</td>
</tr>
<tr>
<td></td>
<td>y: the vertical coordinate of where the text should be placed</td>
<td></td>
</tr>
<tr>
<td>geom_errorbar()</td>
<td>x: the variable to plot ymin, ymax: lower and upper value for error bar</td>
<td>colour, size, linetype, width, alpha</td>
</tr>
<tr>
<td>geom_hline()</td>
<td>yintercept = value</td>
<td>colour, size, linetype, alpha</td>
</tr>
<tr>
<td>geom_vline()</td>
<td>xintercept = value</td>
<td>colour, size, linetype, alpha</td>
</tr>
</tbody>
</table>
**msleep**: a file from *ggplot2*

- It contains sleep habits of different mammal species.

```r
msleep

<table>
<thead>
<tr>
<th></th>
<th>name</th>
<th>genus</th>
<th>vore</th>
<th>order</th>
<th>conservation</th>
<th>sleep_total</th>
<th>sleep_rem</th>
<th>sleep_cycle</th>
<th>awake</th>
<th>brainwt</th>
<th>bodywt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cheetah</td>
<td>Acinonyx</td>
<td>carni</td>
<td>Carnivora</td>
<td>&lt;NA&gt;</td>
<td>12.1</td>
<td>NA</td>
<td>11.9</td>
<td>1.9</td>
<td>NA</td>
<td>50.00</td>
</tr>
<tr>
<td>2</td>
<td>Owl monkey</td>
<td>Actus</td>
<td>omni</td>
<td>Primates</td>
<td>&lt;NA&gt;</td>
<td>17.0</td>
<td>1.8</td>
<td>NA</td>
<td>7.0</td>
<td>0.01550</td>
<td>0.4800</td>
</tr>
<tr>
<td>3</td>
<td>Mountain beaver</td>
<td>Aplodontia</td>
<td>herbi</td>
<td>Rodentia</td>
<td>rt</td>
<td>14.4</td>
<td>2.4</td>
<td>NA</td>
<td>9.6</td>
<td>NA</td>
<td>1.350</td>
</tr>
<tr>
<td>4</td>
<td>Greater short-tailed shrew</td>
<td>Blarina</td>
<td>omni</td>
<td>Soricomorpha</td>
<td>lc</td>
<td>14.9</td>
<td>2.3</td>
<td>0.1333333</td>
<td>9.1</td>
<td>0.00029</td>
<td>0.019</td>
</tr>
<tr>
<td>5</td>
<td>Cow</td>
<td>Bos</td>
<td>herbi</td>
<td>Artiodactyla domesticated</td>
<td>&lt;NA&gt;</td>
<td>4.0</td>
<td>0.7</td>
<td>0.6666667</td>
<td>20.0</td>
<td>0.42300</td>
<td>600.000</td>
</tr>
<tr>
<td>6</td>
<td>Three-toed sloth</td>
<td>Bradypus</td>
<td>herbi</td>
<td>Pilosa</td>
<td>&lt;NA&gt;</td>
<td>14.4</td>
<td>2.2</td>
<td>0.7666667</td>
<td>9.6</td>
<td>NA</td>
<td>3.850</td>
</tr>
</tbody>
</table>
```

```r
msleep %>%
ggplot(aes(sleep_cycle, awake)) +
geom_point() -> example.plot

example.plot
```
General structure

Graphs we are going to plot:

- Scatterplot
- Stripchart
- Histogram
- Boxplot
- Violin/beanplot
- Bar chart
- Line chart
- Stacked bar chart
**msleep**: a file from *ggplot2*

- It contains sleep habits of different mammal species.

```
msleep

<table>
<thead>
<tr>
<th>name</th>
<th>genus</th>
<th>vore</th>
<th>order</th>
<th>conservation</th>
<th>sleep_total</th>
<th>sleep_rem</th>
<th>sleep_cycle</th>
<th>awake</th>
<th>brainwt</th>
<th>bodywt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheetah</td>
<td>Acinonyx</td>
<td>cani</td>
<td>Carnivora</td>
<td>lc</td>
<td>12.1</td>
<td>NA</td>
<td>NA</td>
<td>11.9</td>
<td>NA</td>
<td>50.000</td>
</tr>
<tr>
<td>Owl monkey</td>
<td>Actus</td>
<td>omni</td>
<td>Primates</td>
<td>&lt;NA&gt;</td>
<td>17.0</td>
<td>1.8</td>
<td>NA</td>
<td>7.0</td>
<td>0.01550</td>
<td>0.480</td>
</tr>
<tr>
<td>Mountain beaver</td>
<td>Aploclis</td>
<td>herbi</td>
<td>Rodentia</td>
<td>rt</td>
<td>14.4</td>
<td>2.4</td>
<td>NA</td>
<td>9.6</td>
<td>NA</td>
<td>1.350</td>
</tr>
<tr>
<td>Greater short-tailed shrew</td>
<td>Blarina</td>
<td>omni</td>
<td>Soricomorpha</td>
<td>lc</td>
<td>14.9</td>
<td>2.3</td>
<td>0.1333333</td>
<td>9.1</td>
<td>0.00029</td>
<td>0.019</td>
</tr>
<tr>
<td>Cow</td>
<td>Bos</td>
<td>herbi</td>
<td>Artiodactyla</td>
<td>domesticated</td>
<td>4.0</td>
<td>0.7</td>
<td>0.6666667</td>
<td>20.0</td>
<td>0.42300</td>
<td>600.000</td>
</tr>
<tr>
<td>Three-toed sloth</td>
<td>Bradypus</td>
<td>herbi</td>
<td>Pilosa</td>
<td>&lt;NA&gt;</td>
<td>14.4</td>
<td>2.2</td>
<td>0.7666667</td>
<td>9.6</td>
<td>NA</td>
<td>3.850</td>
</tr>
</tbody>
</table>
```

- Relationship between numbers of hours slept (**sleep_total**) and weight (**bodywt**):
  > Scatterplot

- Numbers of hours slept (**sleep_total**) by trophic levels (**vore**):
  > Stripchart

- To remove the ‘NAs’ from the vore:

```
msleep %>>%
  filter(!is.na(vore)) -> msleep.clean
```
Scatterplot

- Basic command:

```r
ggplot(
  data=msleep.clean,
  aes(x=bodywt, y=sleep_total)
)+geom_point()
```
Scatterplot

- Default theme: `theme_grey` changed to `theme_set()`
- Axis labels: `xlab()` and `ylab()`
- Graph title: `ggtitle()`
- Change ticks and limits for a continuous variable: `scale_y_continuous(breaks=,limits=c())`
  - Can also change axis labels
- Colours = categories/levels of a factor: `colour=factor`
- Change colours: `scale_colour_brewer()`
  - `palette=`
  - Add a legend: `name=, labels=c()`
  - Manually: `scale_colour_manual(values=c())`
Changing scaling

• Alter the data before plotting
  – mutate(value=log(value))

• Alter the data whilst plotting
  – ggplot(aes(log(value)))

• Alter the scale of the plot
  – Add an option to adjust the scaling of the axis
Axis scaling options

• Transforming scales
  – scale_x_log10()
  – scale_x_sqrt()
  – scale_x_reverse()

  Equivalent _y_ versions also exist

• Adjusting ranges
  – scale_x_continuous()
    • limits=c(-5,5)
    • breaks=seq(from=-5,by=2,to=5)
    • minor_breaks
    • labels
  – coord_cartesian()
    • xlim=c(-5,5)
    • ylim=c(10,20)
Plot log data: **log scale** vs. log transformed data

```r
ggplot(
  msleep.clean,
  aes(x=bodywt, y=sleep_total, colour=vore)
)+geom_point() -> scatterplot

scatterplot + scale_x_log10()
```
Scatterplot

Plot log data: log scale vs. log transformed data

```r
ggplot(
  msleep.clean,
  aes(x=log(bodywt), y=sleep_total, colour=vore)
)+geom_point()
```
Scatterplot

Add axis labels and main title

ggplot(
  msleep.clean,
  aes(x=log(bodywt), y=sleep_total, colour=vore)
) +
  geom_point(size=4) +
  xlab("Log Body Weight") +
  ylab("Total Hours Sleep") +
  ggtitle("Some Sleep Data") -> scatterplot
Titles and axis labels

• Can set everything with `labs()`
  – `title="Main title"`
  – `x="X axis"`
  – `y="Y axis"`

• Can use functions to set them individually
  – `ggtitle()`
  – `xlab()`
  – `ylab()`
Scatterplot

Change the general theme and move the main title in the centre

```r
theme_set(theme_bw(base_size=18))

scatterplot+ theme(plot.title = element_text(hjust = 0.5))
```
ggPlot Themes

- theme_grey()
- theme_bw()
- theme_linedraw()
- theme_light()
- theme_dark()
- theme_minimal()
- theme_classic()
Customising themes

• When setting the theme
  – `theme_set(theme_bw(base_size=14))`
  – Only font sizes, family and line weight

• In a plot
  – Only applies to a single plot
  – Use `+theme(plot.title = element_text(hjust=0.5))`

• From this point on
  – Use `theme_update(plot.title = element_text(hjust = 0.5))`
What can you customise?

theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x, axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left, axis.title.y.right, axis.text, axis.text.x, axis.text.x.top, axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right, axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y, axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length, axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y, axis.line.y.left, axis.line.y.right, legend.background, legend.margin, legendspacing, legend.spacing.x, legend.spacing.y, legend.key, legend.key.size, legend.key.height, legend.key.width, legend.text, legend.text.align, legend.title, legend.title.align, legend.position, legend.direction, legend.justification, legend.box, legend.box.just, legend.box.margin, legend.box.background, legend.box.spacing, panel.background, panel.border, panel.spacing, panel.spacing.x, panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor, panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x, panel.grid.minor.y, panel.ontop, plot.background, plot.title, plot.subtitle, plot.caption, plot.tag, plot.tag.position, plot.margin, strip.background, strip.background.x, strip.background.y, strip.placement, strip.text, strip.text.x, strip.text.y, strip.switch.pad.grid, strip.switch.pad.wrap

https://ggplot2.tidyverse.org/reference/theme.html
Scatterplot

Change the general theme and move the main title in the centre

```
theme_set(theme_bw(base_size=18))

scatterplot+ theme(plot.title = element_text(hjust = 0.5))
```
Scatterplot

Change the colours (palette), the legend labels and the ticks on the axes.

```r
scatterplot +
  scale_colour_brewer(
    palette="Set1",
    name="Trophic levels",
    labels=c("Carnivore", "Herbivore", "Insectivore", "Omnivore")
  ) +
  scale_x_continuous(breaks=-5:10) +
  scale_y_continuous(breaks=seq(0,20, 2))
```
Selecting categorical colours

• Modify default colours
  – `scale_fill_hue(c=30, l=40)` (also `scale_colour_hue`)
  – Adjusts luminance and chroma of default colours
  – Makes them less bright (which they are!)

• Use a different scale
  – Existing pre-defined scale
  – Manually created scale
Using a different colour scale

• **ColorBrewerScales**

```r
scale_colour_brewer(
    palette="name",
    labels=c("Override","Default","Labels"),
    name="top title",
)
```

• **ManualScale**

```r
scale_colour_manual(
    values=c("red","green","blue")
)
```
Scatterplot

Change the colours (manual).

```r
scatterplot +
  scale_colour_manual(
    values=c("chocolate3", "chartreuse3", "darkorchid2", "cyan3"),
    name="Trophic levels",
    labels=c("Carnivore", "Herbivore", "Insectivore", "Omnivore")
  ) -> scatterplot
```
Exercise 1: Scatterplot

*up_down_expression.txt*

- The file *up_down_expression.txt* contains an expression comparison dataset with an extra column which classifies the rows into one of 3 groups (up, down or unchanging).

- Load: “up_down_expression.txt“
- Check the structure of the file
- **Plot a scatterplot** with:
  - up in red,
  - down in blue
  - unchanging in grey
  - Main title: “Expression data”
  - Colour legend: “Down, Unchanging and Up”
  - Axis labels: “Condition 1” and “Condition 2”
Exercise 1: Scatterplot

`up_down_expression.txt`

```r
expression <- read_tsv("up_down_expression.txt")
expression

expression.scatter <- ggplot(expression, aes(Condition1, Condition2, colour=State)) +
  geom_point() +
  scale_colour_manual(values=c("blue", "grey", "red"),
                      name="State",
                      labels=c("Down", "Unchanging", "Up")) +
  xlab("Condition 1") +
  ylab("Condition 2") +
  ggtitle("Expression data") +
  theme(plot.title = element_text(hjust = 0.5))

expression.scatter
```
Stripchart

- Basic command:

```r
ggplot(
  msleep.clean,
  aes(vore, sleep_total)
)+ geom_point()
```
Stripchart

- Change ticks and labels for a categorical variable: `scale_x_discrete(labels=c())`
  - Can also change axis labels

- `geom_point(position= "jitter")`
- `geom_jitter()`

- Add segment to visualise means:
  - `stat_summary` sums up data using chosen a function:
    - `stat_summary(fun.y='mean')`: calculates the mean for all the 4 `vore` categories
    - `geom='errorbar'`: choice of graphical representation
      - Default: `geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd))`
  - Cheeky: we only want the means and the segments
    - `aes(ymin=..y.., ymax=..y..)`
ggplot(msleep.clean, aes(vore, sleep_total, colour=vore)) + geom_jitter(width = .2, height = 0, size = 4) -> stripchart
Stat Summary

• Add summary statistics to discrete data

• Main options
  – geom – how is this going to be displayed
    • pointrange (default)
    • errorbar
    • linerange
    • Crossbar

  – fun.data
    • Function to produce
      – Min, Centre, Max
      – Eg mean_se, mean_cl_boot, mean_cl_normal, mean_sdl
    • Can use fun.ymin, fun.y, fun.ymax
stripchart +

stat_summary(
  fun.y=mean,
  fun.ymin=mean,
  fun.ymax=mean,
  geom='errorbar',
  width=0.6,
  size=1.5,
  colour="grey25"
) -> stripchart
stripchart +
  ylab("Total Hours Sleep") +
  xlab("Trophic Levels") +
  ggtitle("Some Sleep Data") +
  scale_y_continuous(breaks=seq(0, 20, 2)) +
  scale_x_discrete(labels=c("Carnivore", "Herbivore", "Insectivore", "Omnivore")) +
  theme(legend.position = "none") -> stripchart
Stripchart

```r
stripchart +
  scale_colour_brewer(palette="Dark2") +
  scale_x_discrete(
    limits=c("insect", "herbi", "carni", "omni"),
    labels=c("Insectivore", "Herbivore", "Carnivore", "Omnivore")) +
  theme(plot.title = element_text(hjust = 0.5))
) -> stripchart
```
"DownloadFestival.csv"

- It contains the hygiene scores (0-5) of 810 concert goers over three days of music festival.

```
head(festival.data)
```
```
ticknmb | gender | day1 | day2 | day3
-------|--------|------|------|------
   1    | Male   | 2.64 | 1.35 | 1.61
   2    | Female | 0.97 | 1.41 | 0.29
   3    | Male   | 0.84 |   NA |   NA
   4    | Female | 3.03 |   NA |   NA
   5    | Female | 0.88 | 0.08 |   NA
   6    | Male   | 0.85 |   NA |   NA
```

Distribution of scores per day and gender:
> Histogram, stripchart, boxplot and violinplot

Means and error bars
> Barchart and linegraph

```
read_csv("DownloadFestival.csv") -> festival.data
```
Histogram

- **Basic command:**

```
ggplot(
    festival.data,
    aes(day1)
)+geom_histogram() -> Day1Histogram
```

```
ggplot(festival.data, aes(day1)) +
    geom_histogram(binwidth=0.3, colour="black", fill="yellow") +
    xlab("Score") +
    ylab("Counts") +
    theme(plot.title = element_text(hjust = 0.5)) +
    ggtitle("Hygiene at Day 1") -> Day1Histogram
```
To plot all 3 days, we need to restructure the file:

```r
festival.data %>%
gather(day, score, -ticknumb, -gender) -> festival.data.stack
```

Remove NA values:

```r
festival.data.stack %>% filter(!is.na(score)) -> festival.data.stack
festival.data.stack
```
Histogram

- To plot several graphs on the same panel: `facet_grid(raw ~ column)`
  - `raw ~ column`: variable names e.g. gender and day

```r
ggplot(festival.data.stack, aes(score)) + geom_histogram(binwidth=0.3, colour="black", fill="yellow") + labs(x="Hygiene score", y="Counts") + facet_grid(gender~day) -> histogram.3days
```
Exercise 2: Stripchart with facet_grid

- Plot a stripchart representing all 3 days and each gender, you will need:
  - `facet_grid(~column)`
  - `scale_colour_manual(values=c("darkorange", "darkorchid4"))`
Exercise 2: Stripchart with facet_grid

ggplot(festival.data.stack, aes(gender, score, colour=gender)) +
  facet_grid(~ day) +
  geom_point(position="jitter") +
  scale_colour_manual(values=c("darkorange", "darkorchid4")) +
  stat_summary(geom='errorbar', fun.y=mean, aes(ymin=..y.., ymax=..y..),
               colour="black", width=0.8, size=1.5) +
  labs(x="Gender", y="Score") +
  theme(legend.position = "none") -> stripchart
Stripchart + any layer

- From a stripchart we can add a line for the mean or any other descriptive geom as a stat summary.

```r
ggplot(festival.data.stack, aes(gender, score, colour=gender)) + facet_grid(~day) + geom_point(position="jitter") + scale_colour_manual(values=c("darkorange", "darkorchid4")) + labs(x="Gender", y="Score") + theme(legend.position = "none") -> stripchart
```
Stripchart + mean

- Add a mean

```r
stripchart+
stat_summary(fun.y="mean", geom="errorbar", aes(ymin=..y.., ymax=..y..), width=0.8, colour="black", size = 1.3)
```
Stripchart + boxplot

- To add a geom as a transparent layer: `alpha=0`

```r
stripchart+
  geom_boxplot(alpha=0, colour="black")
```
Stripchart + boxplot prettier

```r
stripchart+\n  geom_boxplot(aes(gender, score, fill=gender), alpha=0.5, colour="black")+
  scale_fill_manual(values=c("darkorange", "darkorchid4"))
```
Boxplot

- Basic command:

```r
ggplot(festival.data.stack, aes(gender, score)) +
  geom_boxplot() +
  facet_grid(~day) -> boxplot
```
Boxplot

• Prettier:
```r
ggplot(festival.data.stack, aes(gender, score, fill=gender)) +
  facet_grid(~day) +
  stat_boxplot(geom="errorbar", width=0.5) +
  geom_boxplot(outlier.shape=8) +
  theme(legend.position = "none") +
  scale_fill_manual(values=c("sienna1","darkorchid3 ")) +
  labs(x="Gender", y="Hygiene score") -> boxplot
```
Stripchart + violinplot

```r
geom_violin

stripchart+
    geom_violin(alpha=0, colour="black")
```
Violinplot (Beanplot)

- Basic command:

```r
ggplot(festival.data.stack, aes(gender, score)) + geom_violin() + facet_grid(~day)
```
Violinplot (Beanplot)

```r
ggplot(festival.data.stack, aes(gender, score, fill=gender)) +
  facet_grid(~day) +
  geom_violin(trim = FALSE) +
  scale_fill_manual(values=c("goldenrod2","darkgrey")) +
  theme(legend.position="none") +
  stat_summary(fun.y=median, geom="point", size=2, color="black") +
  labs(x="Gender", y="Hygiene scores") -> violinplot
```
Violinplot combination

`violinplot+geom_boxplot(width=0.3)`

`violinplot+geom_jitter(width=0.1, size=1, shape=1)`
Bar chart

• First we want to calculate the mean and sem and store the values in a file.

```r
festival.data.stack %>%
  group_by(gender, day) %>%
  summarise(mean = mean(score), sem = sd(score)/sqrt(n())) -> score.sem
```

```r
table(score.sem)
```

1. Female day1 1.8787273 0.03164061
2. Female day2 1.0828750 0.06077612
3. Female day3 1.0997015 0.09895861
4. Male day1 1.6020635 0.03619580
5. Male day2 0.7732692 0.05847218
6. Male day3 0.8291071 0.07209944

• Basic command:

```r
ggplot(score.sem, aes(day, mean, fill=gender)) +
  geom_bar(stat="identity") -> barchart
```
Bar chart

- **position="dodge"**: elements (e.g. bars) next to each other rather than stacked

- **position=position_dodge(width=0.9)**:
  - the default width for bars is 0.9
  - to narrow the error bars with `geom_errorbar()`, we need to tell `ggplot` to adjust their position to the width of the bar (0.9).

```r
ggplot(score.sem, aes(day,mean, fill=gender)) + geom_bar(stat="identity", position="dodge") + geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem), position=position_dodge(0.9)) -> barchart
```
ggplot(score.sem, aes(day, mean, fill=gender)) +
  geom_bar(position="dodge", colour="black", stat="identity", size=1) +
  geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem), width=.5, position=position_dodge(0.9), size=1) +
  ylab("Mean scores") +
  ggtitle("Levels of hygiene over 3 days of concert") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.title.x = element_blank()) +
  scale_fill_manual(values=c("darkorange3", "darkorchid4"), name="Gender") -> barchart
**Linegraph**

- **Basic command:**

```r
linegraph <- ggplot(score.sem, aes(day, mean, group=gender)) +
  geom_line() +
  geom_point() +
  geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem))
```
Linegraph

• Prettier:
  • Colour per gender: `colour=gender`

```r
ggplot(score.sem, aes(day, mean, colour=gender)) +
  geom_line(size=1.5) +
  geom_point(size=4) +
  geom_errorbar(aes(ymin=mean-sem, ymax=mean+sem), width=.2, size=1.5) -> linegraph
```
Linegraph

- Changing the legend:
  - to change the position of the legend on the graph: `theme(legend.position = c(),)`
  - to change the aspect:
    - `theme(legend.text = element_text(size = , face = ))`
    - `theme(legend.background = element_rect(fill = "transparent"))`

```r
Linegraph+
scale_y_continuous(breaks = seq(0, 2, 0.2)) +
ggtitle("Levels of hygiene over 3 days of concert") +
theme(plot.title = element_text(hjust = 0.5)) +
scale_colour_manual(values = c("purple","darkorange3"), name = NULL) +
theme(legend.position = c(1,1)) +
theme(legend.justification = c(1,1)) +
theme(legend.text = element_text(size = 14)) +
theme(legend.background = element_blank())
```

Levels of hygiene over 3 days of concert
Exercise 3: Linegraph
"chromosome_position_data.txt"

- The file contains positional count data for 3 different datasets (a WT and two mutants).

- **Plot a graph showing all 3 datasets on the same plot**
  - Load: "chromosome_position_data.txt"
  - Check the structure of the file
  - Restructure the file from wide to long format `gather()`
  - Rename the column: "Genotype" and "Value"
  - Plot a basic line graph
Exercise 3: Linegraph
"chromosome_position_data.txt"

```r
chromosome <- read_tsv("chromosome_position_data.txt")

chromosome.long <- melt(chromosome, id="Position")
colnames(chromosome.long)[2:3] <- c("Genotype", "Value")
chromosome.long

chromosome.linegraph <- ggplot(chromosome.long, aes(x=Position, y=Value, group=Genotype, colour=Genotype)) + geom_line(size=2)

chromosome.linegraph
```
Exercise 4: Linegraph "weight_chart.txt"

- Plot a graph showing the relationship between age and weight for a typical baby over the first 9 months.
  - Load: weight_chart.txt
  - Check the structure of the file
  - Plot a basic line graph
  - Plot a prettier version:
    - Change the size and the colour of the points
    - Change the thickness and the colour of the line
    - Change the y-axis: scale from 2 to 10 kgs
    - Change the x-axis: scale from 0 to 10 months
    - Change the labels on both the axis
    - Add a title to the graph
Exercise 4: Linegraph

"weight_chart.txt"

```r
weight <- read_tsv("weight_chart.txt")
weight

## basic ##
ggplot(weight, aes(Age, Weight)) +
  geom_line() +
  geom_point()
```
Exercise 4: Linegraph "weight_chart.txt"

```r
## pretty ##
ggplot(weight, aes(Age, Weight)) +
  geom_line(size=1, colour="lightblue2") +
  geom_point(shape=16, size=3, colour="darkorchid1") +
  scale_y_continuous(breaks=2:10, limits = c(2, 10)) +
  scale_x_continuous(breaks=0:10, limits = c(0, 10)) +
  labs(x="Age (months)", y="Weight (kg)") +
  ggtitle("Relation between age and weight") +
  theme(plot.title = element_text(hjust = 0.5)) -> weight.linegraph
```

![Relation between age and weight](image)
Exercise 5 “brain_bodyweight.txt”

- The file brain_bodyweight.txt contains data for the log10 brain and bodyweight for a range of species, along with an SEM measure for each point.

- Plot these data on a scatterplot with error bars showing the mean +/- SEM and the names of the datasets under each point.
  
  o Load: brain_bodyweight.txt
  o Check the structure of the file
  o Plot a basic graph. You will need:
    - `geom_barh()` for the horizontal error bars
    - `geom_text()` for the labels

  o Plot a prettier version:
    - Something like that
Exercise 5 "brain_bodyweight.txt"

```r
## basic ##
ggplot(brain.bodyweight, aes(x=Bodyweight, y=Brainweight)) +
geom_point() +
geom_errorbar(aes(ymin=Brainweight-Brainweight.SEM, ymax=Brainweight+Brainweight.SEM)) +
geom_errorbarh(aes(xmin=Bodyweight-Bodyweight.SEM, xmax=Bodyweight+Bodyweight.SEM)) +
geom_text(aes(label=Species), hjust = 1.05, vjust = -0.6, size=2.7) ->
brain.bodyweight.graph
```
Exercise 5 “brain_bodyweight.txt”

```r
## pretty ##
ggplot(brain.body, aes(x=Bodyweight, y=Brainweight)) +
  geom_errorbar(aes(ymin=Brainweight-Brainweight.SEM, ymax=Brainweight+Brainweight.SEM),
                width=.1, size=1, colour="tomato3") +
  geom_errorbarh(aes(xmin=Bodyweight-Bodyweight.SEM, xmax=Bodyweight+Bodyweight.SEM),
                 height=.1, size=1, colour="tomato3") +
  geom_point(size=3) +
  geom_text(aes(label=Species), hjust = 1.1, vjust = -0.6, size=2.7) ->
brain.bodyweight.graph
```
# prettier with a new package: ggrepel#

```r
library("ggrepel")

ggplot(brain.bodyweight, aes(x=Bodyweight, y=Brainweight)) +
  geom_errorbar(aes(ymin=Brainweight-Brainweight.SEM, ymax=Brainweight+Brainweight.SEM),
                width=.1, size=0.5, colour="grey28") +
  geom_errorbarh(aes(xmin=Bodyweight-Bodyweight.SEM, xmax=Bodyweight+Bodyweight.SEM),
                 height=.1, size=0.5, colour="grey28") +
  geom_point(shape=21, size=3, colour="black", fill="maroon3") +
  geom_label_repel(aes(label = Species), box.padding=0.6, point.padding =0.5,
                   fill="mintcream", segment.colour="grey", size=3) -> brain.bodyweight.graph
```

Exercise 5 “brain_bodyweight.txt”
"Changing.csv"

- It contains data from 12 problem behaviours (`Type.of.Behaviour`) and the different stages (5) of change subjects are going through (`Stage.of.Change`).

```r
Changing <- read_csv("Changing.csv")

# A tibble: 60 x 3

`Type of Behaviour` `Sample Size` `Stage of Change`  
<chr>         <dbl>            <chr>          
1 Smoking cessation  108 Precontemplation
2 Smoking cessation  187 Contemplation
3 Smoking cessation   0 Preparation
4 Smoking cessation  134 Action
5 Smoking cessation  247 Maintenance
6 Quitting cocaine    8 Precontemplation
7 Quitting cocaine   15 Contemplation
8 Quitting cocaine    0 Preparation
9 Quitting cocaine   71 Action
10 Quitting cocaine   62 Maintenance
# ... with 50 more rows
```

Colnames aren’t ‘fixed’ by read_csv so you need backtick quotes around them when you use them. You can ‘fix’ them by using:

```r
colnames(changing) <- make.names(colnames(changing))
```
Stacked Bar Chart (raw data)

- Basic command:

```r
ggplot(Changing, aes(Type.of.Behaviour, Sample.Size, fill=Stage.of.Change)) + geom_bar(stat="identity") -> stackedBar
```

- Changing the order of the levels: `factor(variable name, levels = c("", "" ...))`

- Rotate the graph to read the x-axis labels: `coord_flip()`
Stacked Bar Chart (raw data)

```r

ggplot(Changing, aes(Type.of.Behaviour, Sample.Size, fill = Stage.of.Change)) +
  geom_bar(stat="identity", colour="black") +
  coord_flip() -> stackedBar
```
Stacked Bar Chart (raw data)

```r
codedBar+
  theme(plot.title = element_text(hjust = 0.5, size=12, face="bold"))+
  theme(axis.title.y = element_blank())+
  scale_fill_brewer(palette = 4)+
  theme(axis.text.x = element_text(size=10), axis.text.y = element_text(size=9))+
  theme(legend.text = element_text(size=8), legend.title = element_text(size=10, face="bold"))+
  theme(axis.title.x = element_text(size=10))
```

**Stages for Each of the 12 Problem Behaviours**

- Weight control
- Sunscreen use
- Smoking cessation
- Safer sex
- Radon gas exposure
- Quitting cocaine
- Physicians' practices
- Mammography screening
- High fat diet
- Exercise acquisition
- Condom use
- Adolescent delinquency

```
```

**Legend**

- Maintenance
- Action
- Preparation
- Contemplation
- Precontemplation

**Sample Size**

0 100 200 300 400 500 600 700

```
```
Stacked Bar Chart (raw data)

- Divergent palette to emphasize effect:

scale_fill_brewer(palette="RdYlGn", direction=-1)
Exercise 6: Stacked Bar Chart (percentage data)

- Plot the *Changing* data as percentages.
  - Change the data to be expressed as percentages using *dplyr*
  - Group by “Type of Behaviour”
  - Run *summarise* to get the *sum* of each Behaviour
  - Do a *right_join* of this *total* back to the original data
  - Use the *total* with *mutate* to get the percentage of each count

NB We could achieve a similar effect by adding *position*=`position_fill()` in *geom_bar()`
Exercise 6: Stacked Bar Chart (percentage data)

```r
changing %>%
  group_by(`Type of Behaviour`) %>%
  summarise(total=sum(`Sample Size`)) %>%
  right_join(changing) %>%
  mutate(percent=100*`Sample Size`/total)
```

# A tibble: 60 x 5

```
  `Type of Behaviour` total `Sample Size` `Stage of Change` percent
  <chr>        <dbl>         <dbl>          <chr>      <dbl>
1  Smoking cessation  676          108 Precontemplation 16.0
2  Smoking cessation  676          187 Contemplation  27.7
3  Smoking cessation  676           0 Preparation          0
4  Smoking cessation  676          134 Action  19.8
5  Smoking cessation  676         247 Maintenance  36.5
6   Quitting cocaine  156           8 Precontemplation  5.13
7   Quitting cocaine  156          15 Contemplation  9.62
8   Quitting cocaine  156          0 Preparation          0
9   Quitting cocaine  156          71 Action  45.5
10  Quitting cocaine  156          62 Maintenance  39.7
# ... with 50 more rows
```
Exercise 6: Stacked Bar Chart (percentage data)

ggplot(Changing.percent,aes( Type.of.Behaviour, percent, fill = Stage.of.Change))+
  geom_bar(stat="identity",colour="black")+
  coord_flip()+
  scale_fill_brewer(palette = "Spectral", direction=-1)+
  labs(title="Stages for Each of the 12 Problem Behaviours", y="Frequencies")+
  theme(axis.title.y=element_blank())+
  theme(plot.title = element_text(hjust = 0.5, size=12, face="bold"))+
  theme(axis.text.x = element_text(size=10), axis.text.y = element_text(size=9))+
  theme(legend.text=element_text(size=8),legend.title=element_text(size=10,face="bold"))+
  theme(axis.title.x = element_text(size=10)) -> stackedBar.percent