ggplot2: grammar of graphics

made by a kiwi

Grammar of Graphics (ggplot2)

- Data: This comprises variables that we wish to represent (the relations between).
- Geoms: Geometrical objects such as bars in a barchart will represent the data.
- Aesthetics: These will determine how geoms are drawn
  - A point's position or a histogram bar's shape
- Mappings: from data to aesthetics
  - relates a data value to a point's position
- Scales: will determine how mapping work
  - a larger values makes a taller bar
- Guides: will assist the reader to understand the mappings.
  - tick marks and labels on an axis

we must add them all together

ggplot2

Do it yourself: here we exploit the aggregation abilities of the plyr package to create a summary dataframe:

```
# install.packages('ggplot2')↵
# library(ggplot2)↵
```

Our first plot:

```
> ggplot(mtcars, aes(x=gear, y=mpg)) +
stat_summary(fun.y="mean", geom="bar")↵
```

Notice something? We didn't have to get our hands dirty with *apply! We just had to think about variables.

Building plots (base package)

Building plots is difficult with the base package because every plot is unique (no grammar).

Do it yourself: start by recreating an object we used before:

```
> counts <- table(mtcars$vs, mtcars$gear,
dnn=list("vs","gear"))↵
> counts↵
```

Plotting:

```
> barplot(counts, beside=TRUE, legend=TRUE) ↵
```

How could we group by engine type (vs) instead of gear?

Building plots 2

It requires transposing the data:

Do it yourself:

```
> t(counts) ↵
```

Plotting:

```
> barplot(t(counts), beside=T, legend=T) ↵
```

If we wanted a different set of gems, then we need a new function, e.g., for line graphs:

Do it yourself:

```
> plot(counts[1,], type="l")↵
> lines(counts[2,], col="blue")↵
```

The second data set is plotted outside the plot area.

With ggplot2

For example, for line graphs:

Do it yourself: Let's start by obtaining the data in long format (= one row per entry):

```
> counts <- data.frame(table(mtcars$vs,
mtcars$gear, dnn=list("vs","gear"))) ↵
```

Here is a line plot now:

```
> ggplot(counts, aes(x=factor(gear), y=Freq,
colour=factor(vs), group=factor(vs))) +
geom_line()
```

Changing the geom

For example, for line graphs:

Do it yourself: And here is a bar plot:

```
> ggplot(counts, aes(x=factor(gear), y=Freq,
fill=factor(vs))) +
geom_bar(stat="identity", position="dodge")
```

Plotting Bar Charts

We can use a tidyverse workflow to make an object containing summary data including error bars to plot:

Do it yourself: And here is a bar plot:

```
> ( mtsum <- mtcars %>%
+ group_by(gear) %>%
+ summarise(mmpg=mean(mpg),
ci=qnorm(0.975)*sd(mpg)/sqrt(length(mpg))
+ ))
```

Barplot with Error Bars

Do it yourself: let's plot our aggregated data with error bars all in one go:

```
> ggplot(mtsum, aes(x=gear, y=mmpg)) +
> geom_bar(stat="identity", colour="black",
fill="mistyrose") +
> geom_errorbar(aes(ymin=mmpg-ci,
ymax=mmpg+ci), width=0.2)↵
```

Two Factor Boxplot

Do it yourself: re-assign the mtsum object to crossways data frame:

```
> ( mtsum <- mtcars %>% group_by(gear,vs) %>%
summarise(mmpg=mean(mpg)) ) ↵
```

Now let's plot:

```
> ggplot(mtsum, aes(x=gear, y=mmpg, fill=vs))+
> geom_bar(stat="identity",position="dodge",
colour="black") ↵
```

The scale (automatically added) for vs is weird – it thinks it is a continuous variable!

Resolving Factors

Do it yourself: ok let's tell it that vs is a factor:

```
> ggplot(mtsum, aes(x=gear, y=mmpg,
fill=factor(vs))) +
> geom_bar(stat="identity", position="dodge",
colour="black")↵
```

Don't like the colours? Repeat the above with:

```
+ scale_fill_brewer(palette="Pastel1")
```

Suggestion: try removing the `position="dodge"` argument. What happens?

Reading Recommendations

If you like ggplot2 try this book (one of O'Reilly animal series):

- Chang, W. (2012). *R graphics cookbook*. O'Reilly Media, Inc.

(This is also attribution for the next slide)

- *For more theoretical take see this book.*

Also keep an eye out for developments (@hadleywickham on Twitter and ggvis package).

Scatter Plot

Do it yourself: go get the data:

```
> install.packages('gcookbook')↵
> library(gcookbook)↵
> head(heightweight)↵
```

Basic plot (=recipe 5.1):

```
> ggplot(heightweight, aes(x=ageYear,
y=heightIn)) + geom_point()↵
```

More advanced (=recipe 5.2):

```
> ggplot(heightweight, aes(x=ageYear,
y=heightIn, colour=sex)) + geom_point() +
scale_colour_brewer(palette="Set1") ↵
```

Other Tidyverse Plots: Violin Plots

Do it yourself: basic violin plots:

```
> ggplot(mtcars, aes(x=factor(am), y=mpg)) +
> geom_violin() ↵
```

More advanced violin plot (adapted from recipe 6.9):

```
> ggplot(mtcars, aes(x=factor(am), y=mpg)) + ↵
> geom_violin(trim=FALSE) + ↵
> geom_boxplot(width=.1, fill="black",
outlier.colour=NA) + ↵
> stat_summary(fun.y=median, geom="point",
fill="white", shape=21, size=2.5) ↵
```

Other Tidyverse Plots: Histograms

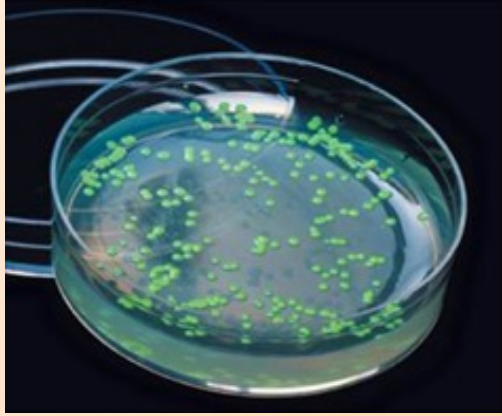
Do it yourself: basic histogram:

```
> ggplot(mtcars, aes(x=mpg)) +
> geom_histogram(fill=”red”, colour="black") ↵
```

More advanced histogram:

```
> money=c(rnorm(10000,2,2),rnorm(10000,5,1)) ↵
> mood=factor(rep(c("love","hate"),each=10000)) ↵
> TB <- tibble(money, mood) ↵
> ggplot(TB, aes(x=money, fill=mood)) + ↵
> geom_histogram(position="identity", alpha=0.4,
colour="black") ↵
```

GFP Example



This example comes from a class of mine.

We measured gene expression using green fluorescent protein. This was pilot data to test the effect of a "photomultiplier" setting.

This shows how to handle multiple predictors using:

- interaction fill,
- facets, which allow alignment of equivalent plots.

It also shows:

- progressive construction and decoration of plots,
- management of devices (reminder).

Facets (GFP Example)

Do it yourself:

```
> gfp <- read.csv("gfp_test_data.csv")
> pdf(file="GFP_bar.pdf", 8, 16) ↵
> basic <- ggplot(gfp, aes(x=factor(time), y=fluor,
fill=interaction(ARA, ethanol))) ↵
> basic <- basic + geom_bar(colour="black",
stat="identity", position="dodge") ↵
> basic <- basic + geom_text(aes(label=fluor),
vjust=1.5, colour="white",
position=position_dodge(.9), size=2.5) ↵
> basic <- basic + facet_grid(gain~., scales="free")
basic + scale_fill_manual(values=c("royalblue",
"darkorange", "maroon")) ↵
> dev.off() ↵
```

Saving in ggplot2

Do it yourself:

```
> ggsave("myplot.pdf") ↵
```

Default behaviour: saves last plot made with ggplot2

Learn more:

```
> ?ggsave ↵
```