chris — -bash — 200×46

```
console
C tr -c "[:digit:]" " " < /dev/urandom | dd cbs=$COLUMNS conv=unblock | GREP_COLOR="1;32" grep --color "[^ ]"
```

Tidyverse

A.K.A How to Kon-Mari your data

Christina Van Heer
christinavanheer@gmail.com

Disclaimers

- I am not an expert in coding - I don't have a background in computer science or engineering where coding is routine

- The advantage (?) of this is that I'm pretty good at knowing what it's like to be like this guy ->

- I learnt to use the tidyverse a few months ago (actual lol) so who knows if I know anything really…..

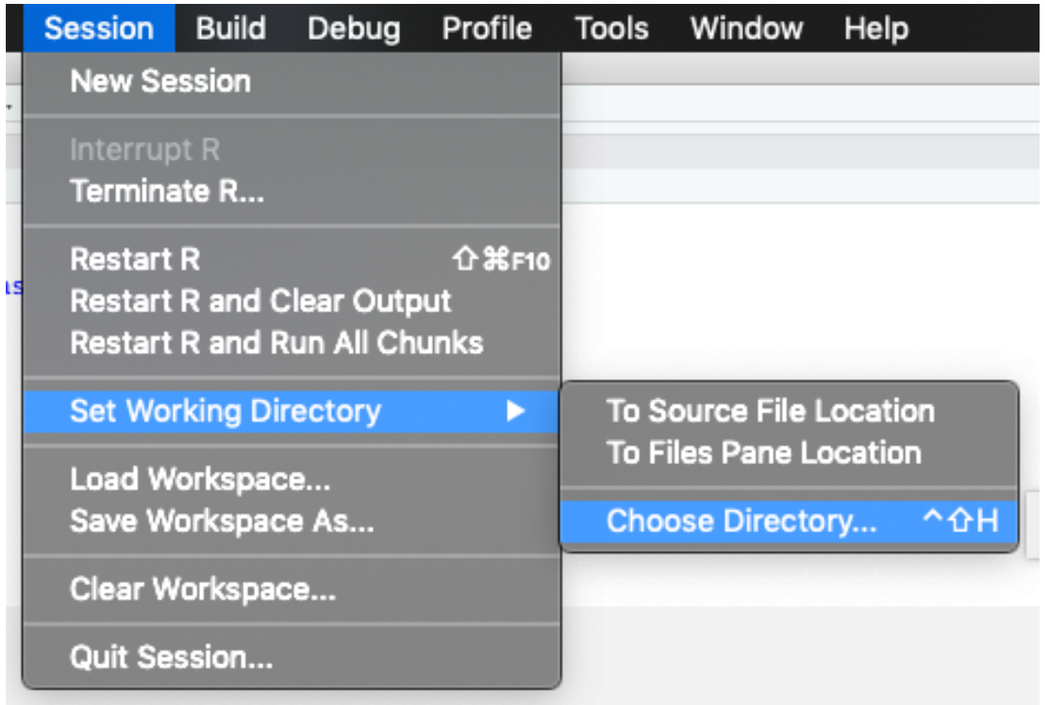
Make sure your R is looking in the right place!

1. Open the R Markdown document you are working with

2. Set your working directory (the place where R studio looks for files)

<u>FAQs</u>

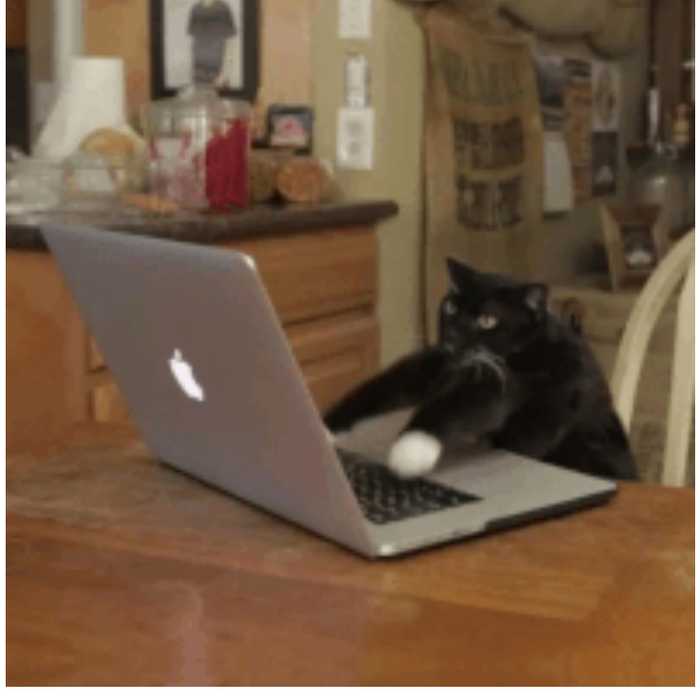
- What is a working directory?
  - It is the place R looks when it is loading stuff
  - So if you have opened an R Markdown file and you want to load some data, R studio needs to be looking in this directory
- How do I set my working directory?
  - Session > Set Working Directory > Choose Directory
    - Make sure you are in the right FOLDER (you don't choose a file) and click ok if you are in the right place



OR

- Session > Set Working Directory > To Source File Location
  - Will do this to the same place where your R markdown file is….but the data needs to be in the SAME folder)

Using R markdown

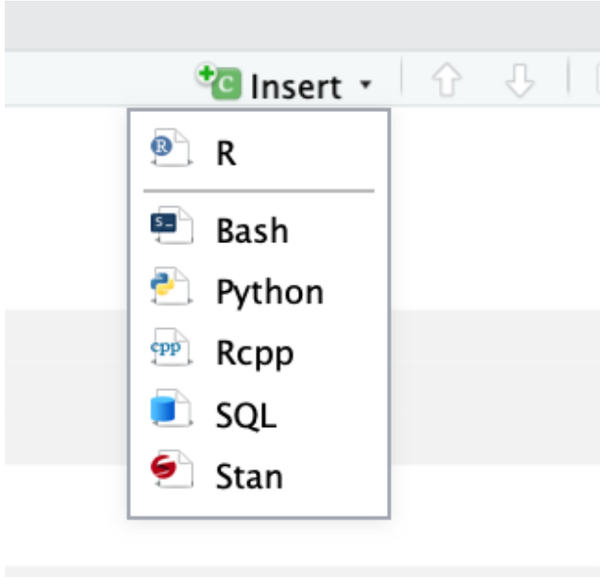


- We want to have a record of what we do
  - A markdown document can do this
  - A markdown document consists of two main areas
    - Code areas (grey bits)
    - Comment / writing areas (white bits)

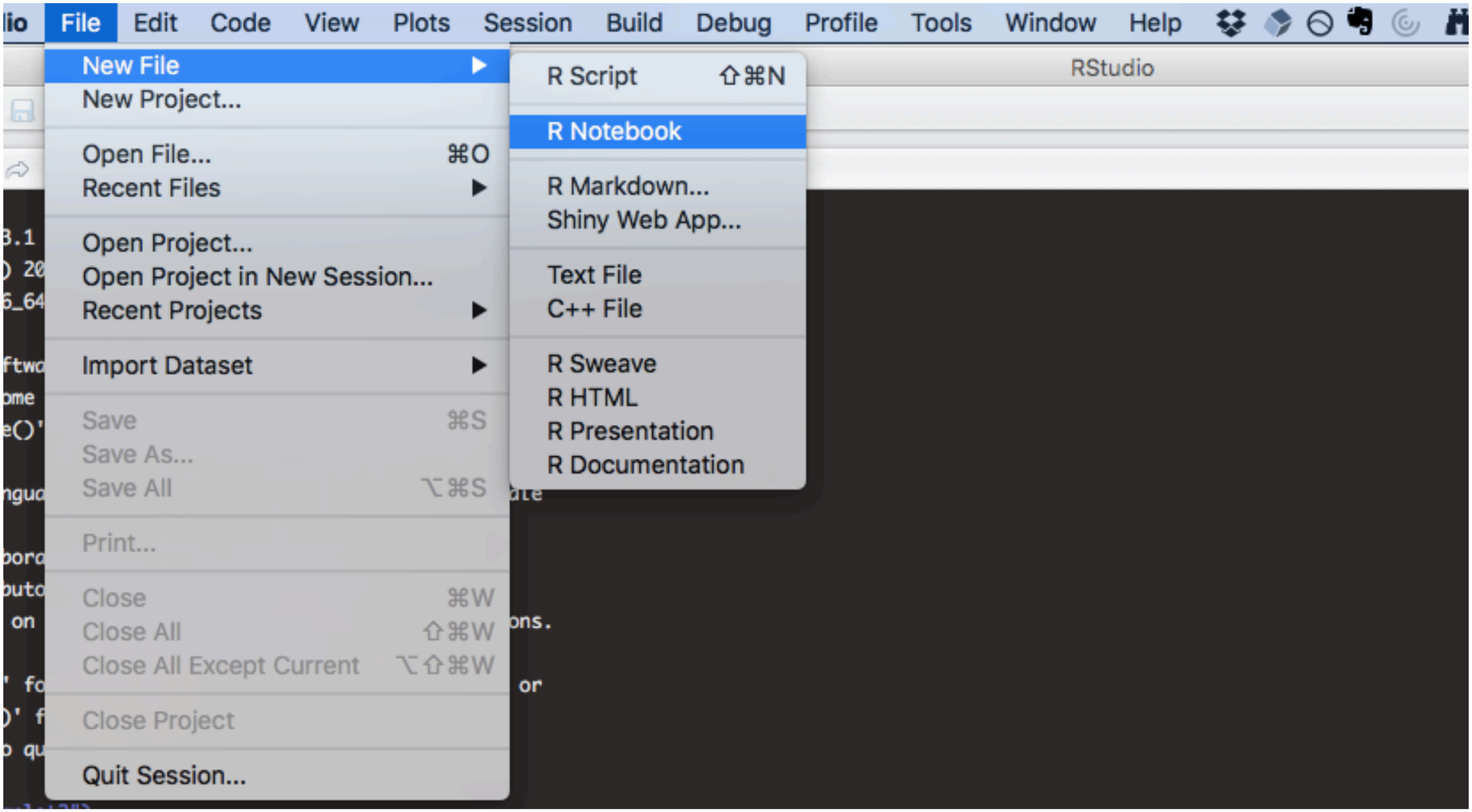
```{r}

```

R studio reads this section as *"oh hey this is code!" This is because* ```*{r}* *is at the start and* ``` *is at the end*

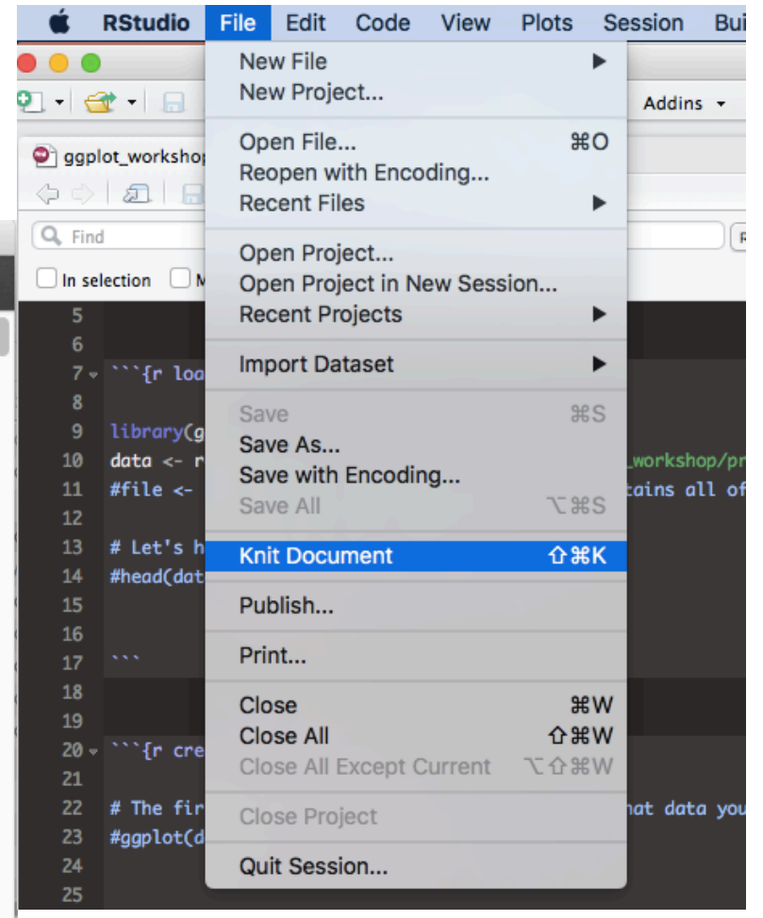


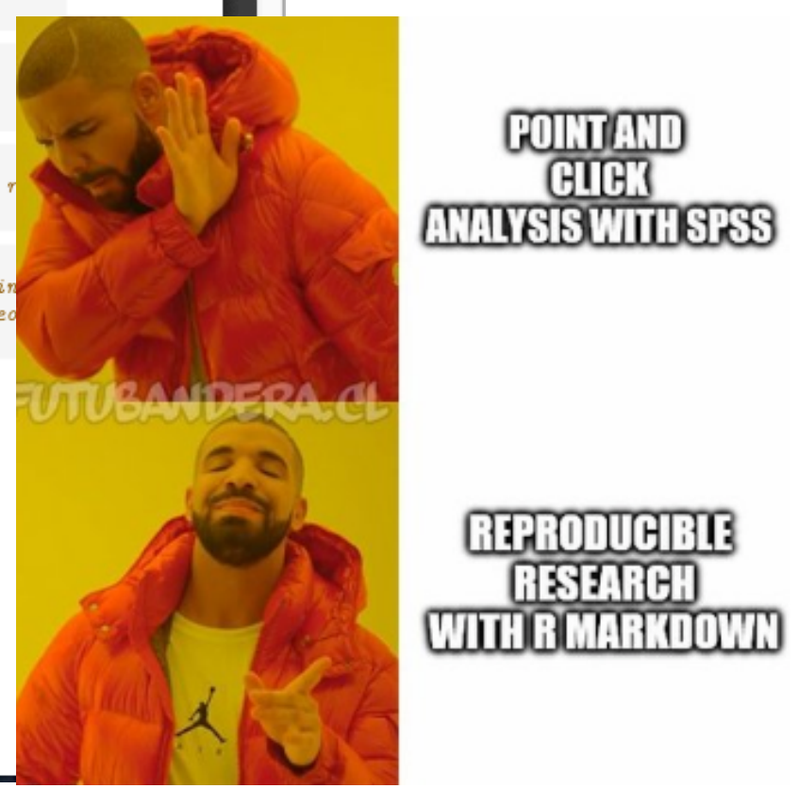
Opening a notebook file



Knit it good

RStudio: View PDF
Page: 1 of 25
Automatic Zoom
ggplot for beginners
library(ggplot2)
Warning: package 'ggplot2' was built under R version 3.3.2
data <- read.csv("~/Dropbox/GRIPS/PhD_camp/ggplot_workshop/processed_data.csv", header=TRUE)
#file <- Sys.glob("*data.csv")[1] # this file contains all of raw data for all subjects in one file
Let's have a quick look at our data
#head(data)
The first thing you need to do is tell ggplot what data you want to plot
#ggplot(data)
Step1 - Tell ggplot what bits of the data you want to plot ggplot data x = trial, y = RT_daty1
Step 2 - Tell ggplot what you want to display it as points
Step 3 - Let's try adding some colour
data[,'conf'] <- as.factor(data[,'conf'])
# geom_point inherits everything from ggplot(data) part
# What if I would like a bar plot?
# Imagine I want to know what the distributions of responses according to confidence
# ggplot(data, aes(x = resp)) + geom_bar(aes(conf))
# We can also change how thin or thick the bars are - the thinness or thickness
# tells us how many data points we count (get frequencies) over. These are called "bin
# Below we can change the bin size to different values using 'binwidth' inside the geo
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.2)


RStudio
File
Edit
Code
View
Plots
Session
New File
New Project...
Open File... ⌘O
Reopen with Encoding...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save ⌘S
Save As...
Save with Encoding...
Save All ⌥⌘S
Knit Document ⇧⌘K
Publish...
Print...
Close ⌘W
Close All ⇧⌘W
Close All Except Current ⌥⇧⌘W
Close Project
Quit Session...
Addins
Find
In selection


POINT AND CLICK ANALYSIS WITH SPSS
REPRODUCIBLE RESEARCH WITH R MARKDOWN

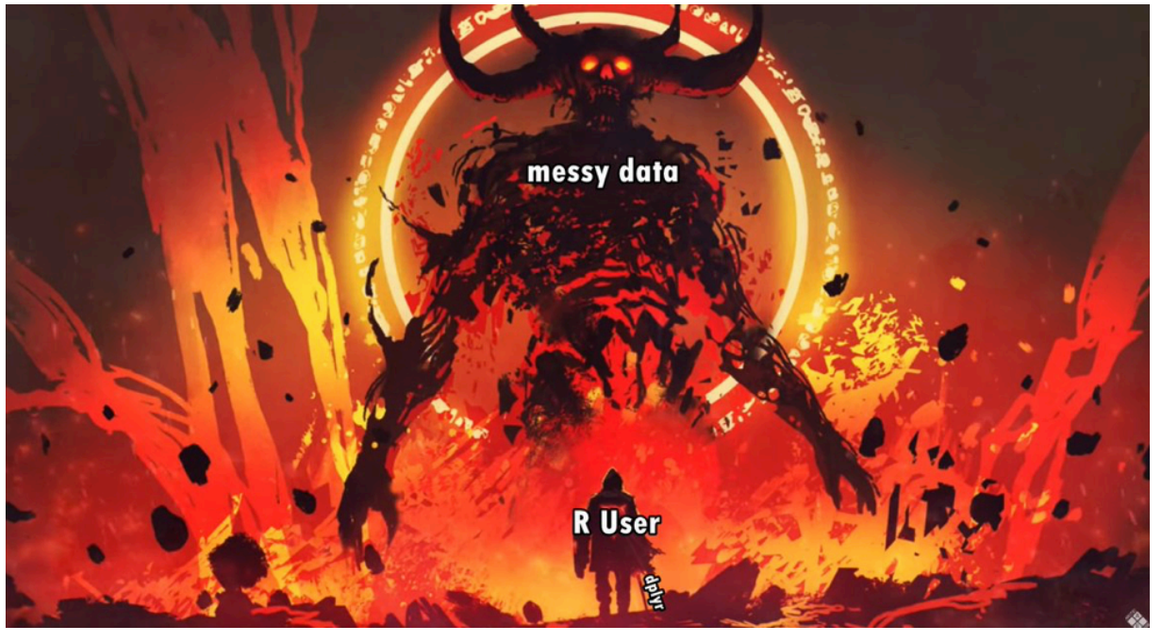
We can change the type of output

```
1  ---
2  title: "R Notebook"
3  output: pdf_notebook
4  ---
5
6  This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.
7
8  Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.
9
10 ```{r}
11 plot(cars)
12 ```
13
14 Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.
15
16 When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press
   *Cmd+Shift+K* to preview the HTML file).
17
```

What is the tidyverse?

- In R, we use functions to perform operations on data
  - You can think of a function as a box that takes something in, performs some operation on it (e.g. calculate the mean) and then spits something out (e.g. the mean)
  - e.g try - mean(c(10, 30, 25, 16) in your R console

- Manipulating datasets in R can be difficult because the way different functions work



- The idea behind tidyverse is that you have a set of packages which contain useful functions to help you manipulate data easily

- The packages work well with each other and allow you to shift the data between them easily

Learning the ways of the Tidyverse (force)

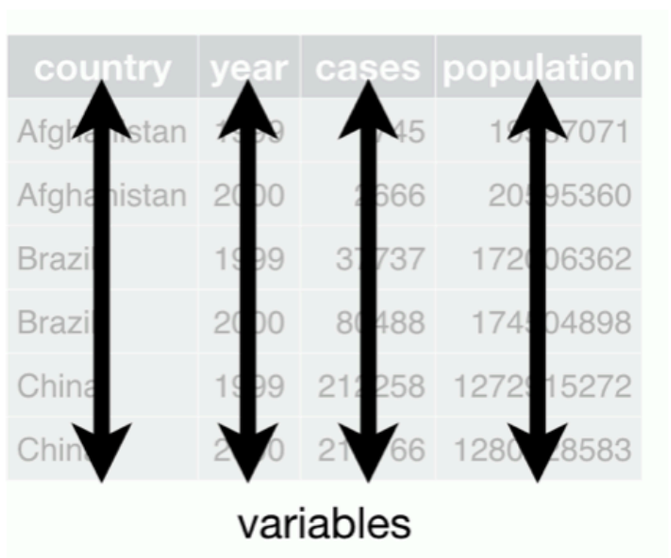
Tibbles vs data frames

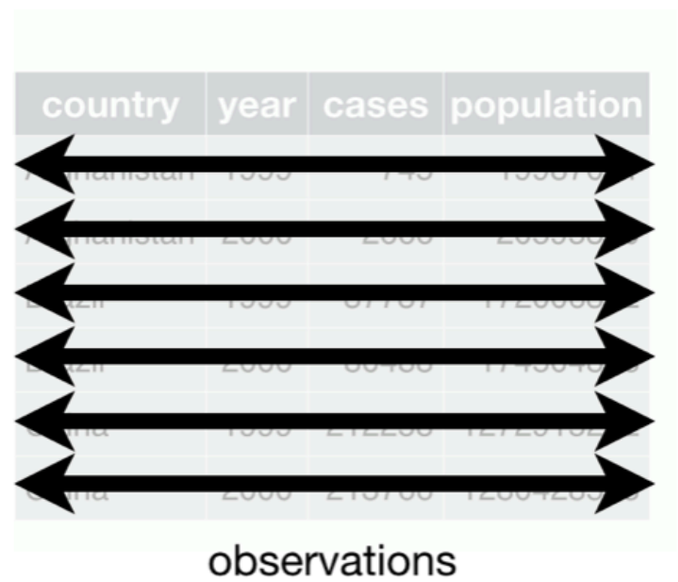
Learning the ways of the Tidyverse (force)

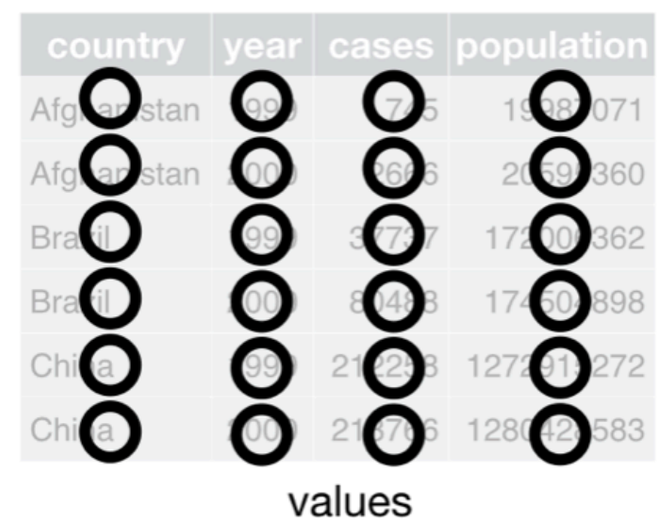
1. Each variable must have its own column.

2. Each observation must have its own row.

3. Each value must have its own cell.







Learning the ways of the Tidyverse (force) - pipes

%>% = a pipe

```
le_data %>%

    do_something %>%

    do_something_else_plz %>%

    also_this_thx
```

Learning the ways of the Tidyverse (force) - group_by

- group_by selects rows out by a particular group
- Useful if you want to do something by subject no. e.g. calculate mean of some measure
- We usually use group_by and then pipe the data onto another function with %>%

```
gapminder %>%
  group_by(country, year)
```

1-10 of 1,704 rows

country	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>	gdpPercap <dbl>
Afghanistan	Asia	1952	28.80100	8425333	779.4453
Afghanistan	Asia	1957	30.33200	9240934	820.8530
Afghanistan	Asia	1962	31.99700	10267083	853.1007
Afghanistan	Asia	1967	34.02000	11537966	836.1971
Afghanistan	Asia	1972	36.08800	13079460	739.9811
Afghanistan	Asia	1977	38.43800	14880372	786.1134
Afghanistan	Asia	1982	39.85400	12881816	978.0114
Afghanistan	Asia	1987	40.82200	13867957	852.3959
Afghanistan	Asia	1992	41.67400	16317921	649.3414
Afghanistan	Asia	1997	41.76300	22227415	635.3414

1–10 of 1,704 rows

Previous 1 2 3 4 5 6 ... 100 Next

Learning the ways of the Tidyverse (force) - arrange

- arrange sorts data out by a particular group
- Useful if you want to view your data in a particular way
- Putting a ‘-‘ sign = descending

```
gapminder %>%
  arrange(-year)
```

```
gapminder %>%
  arrange(year)
```

```
gapminder %>%
  arrange(desc(pop))
```

country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>
Afghanistan	Asia	1952	28.80100	8425333
Albania	Europe	1952	55.23000	1282697
Algeria	Africa	1952	43.07700	9279525
Angola	Africa	1952	30.01500	4232095
Argentina	Americas	1952	62.48500	17876956
Australia	Oceania	1952	69.12000	8691212
Austria	Europe	1952	66.80000	6927772
Bahrain	Asia	1952	50.93900	120447
Bangladesh	Asia	1952	37.48400	46886859
Belgium	Europe	1952	68.00000	8730405

Learning the ways of the Tidyverse (force) - filter

- filter selects data which fulfils a particular rule
- Useful if you want to do something to a selection of the data
- We usually use filter and then pipe the data onto another function with %>%

```
gapminder %>%
  filter(continent == “Europe”)
```

```
gapminder %>%
  filter(year > 1975)
```

```
gapminder %>%
  filter(between(year, 1950, 1975))
```

Learning the ways of the Tidyverse (force) - filter - video

```
  filter(continent == "Europe")

gapminder %>%
  arrange(continent, country)

gapminder %>%
  select(year, lifeExp) %>%
  head(4)

gapminder %>%
  filter(continent == "Europe")

gapminder %>%
  filter(between(year, 1950, 1975))

gapminder %>%
  filter(year > 1975)

```

0 rows

```
# Nesting data
```{r}
gapminder.byCountry <- gapminder %>%
  group_by(country, continent) %>%
  nest()

gapminder.byCountry
```
```

122:14 Chunk 7

Console Terminal R Markdown

```
gapminder %>%
  filter(continent == “Europe”)
```

```
gapminder %>%
  filter(year > 1975)
```

```
gapminder %>%
  filter(between(year, 1950, 1975))
```

Learning the ways of the Tidyverse (force) - select

- Select allows you to select or get rid of specific columns of a tibble
- Useful if you want to get rid of some columns
- We usually use select and then pipe the data onto another function with %>%

```
gapminder %>%
  select(continent, lifeExp)
```

continent <fctr>	lifeExp <dbl>
Asia	28.80100
Asia	30.33200
Asia	31.99700
Asia	34.02000
Asia	36.08800
Asia	38.43800
Asia	39.85400
Asia	40.82200
Asia	41.67400
Asia	41.76300

```
gapminder %>%
  select(-continent, -lifeExp)
```

country <fctr>	year <int>	pop <int>	gdpPercap <dbl>
Afghanistan	1952	8425333	779.4453
Afghanistan	1957	9240934	820.8530
Afghanistan	1962	10267083	853.1007
Afghanistan	1967	11537966	836.1971
Afghanistan	1972	13079460	739.9811
Afghanistan	1977	14880372	786.1134
Afghanistan	1982	12881816	978.0114
Afghanistan	1987	13867957	852.3959

Learning the ways of the Tidyverse (force) - nest()

country <fctr>	continent <fctr>	year <int>	lifeExp <dbl>	pop <int>
Afghanistan	Asia	1952	28.80100	8425333
Afghanistan	Asia	1957	30.33200	9240934
Afghanistan	Asia	1962	31.99700	10267083
Afghanistan	Asia	1967	34.02000	11537966
Afghanistan	Asia	1972	36.08800	13079460
Afghanistan	Asia	1977	38.43800	14880372
Afghanistan	Asia	1982	39.85400	12881816
Afghanistan	Asia	1987	40.82200	13867957
Afghanistan	Asia	1992	41.67400	16317921
Afghanistan	Asia	1997	41.76300	22227415

```
gapminder %>%
  group_by(country, continent) %>%
  nest()
```

country <fctr>	continent <fctr>	data <list>
Afghanistan	Asia	<tibble>
Albania	Europe	<tibble>
Algeria	Africa	<tibble>
Angola	Africa	<tibble>
Argentina	Americas	<tibble>
Australia	Oceania	<tibble>
Austria	Europe	<tibble>
Bahrain	Asia	<tibble>
Bangladesh	Asia	<tibble>

Learning the ways of the Tidyverse (force) - summarize

```
gapminder.byYear <- gapminder %>%
  group_by(year) %>%
  nest()

gapminder.byYear <- gapminder.byYear %>%
  group_by(year) %>%
  unnest(data) %>%
  summarize(meanPop = mean(pop)) %>% # calculate the mean
  inner_join(gapminder.byYear, by = “year)
```

year <fctr>	meanPop <dbl>	data <list>
1952	16950402	<tibble>
1957	18763413	<tibble>
1962	20421007	<tibble>
1967	22658298	<tibble>
1972	25189980	<tibble>
1977	27676379	<tibble>
1982	30207302	<tibble>
1987	33038573	<tibble>
1992	35990917	<tibble>
1997	38839468	<tibble>

Learning the ways of the Tidyverse (force) - linear models

```
gapminder2.byContinent <- gapminder2 %>%
  group_by(continent) %>%
  nest()

gapminder2.byContinent  <- gapminder2.byContinent %>%
  mutate(fit = map(data, ~lm(gdpPercap ~ lifeExp + pop, data = .)))

gapminder2.byContinent
```

continent <fctr>	data <list>	fit <list>
Asia	<tibble>	<S3: lm>
Africa	<tibble>	<S3: lm>
Americas	<tibble>	<S3: lm>
Europe	<tibble>	<S3: lm>
Oceania	<tibble>	<S3: lm>

Learning the ways of the Tidyverse (force) - linear models

country <fctr>	continent <fctr>	data <list>	fit <list>	std.beta <list>	preds <list>	summary.stats <list>
Afghanistan	Asia	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Angola	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Bangladesh	Asia	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Benin	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Burkina Faso	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Burundi	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Cambodia	Asia	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Cameroon	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Central African Republic	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>
Chad	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<tibble>

- Tidyverse becomes hugely powerful when you use it to model your data
- You can store the data that you used to perform an analysis (data), the model you ran on it (fit) the predictions of the model (preds), summary stats like AIC, BIC, R2 etc. (summary.stats)
- Useful if you have multiple experiments, multiple models etc.
- Pipe %>% directly from the tibble with this info into ggplot2 (more detail later)

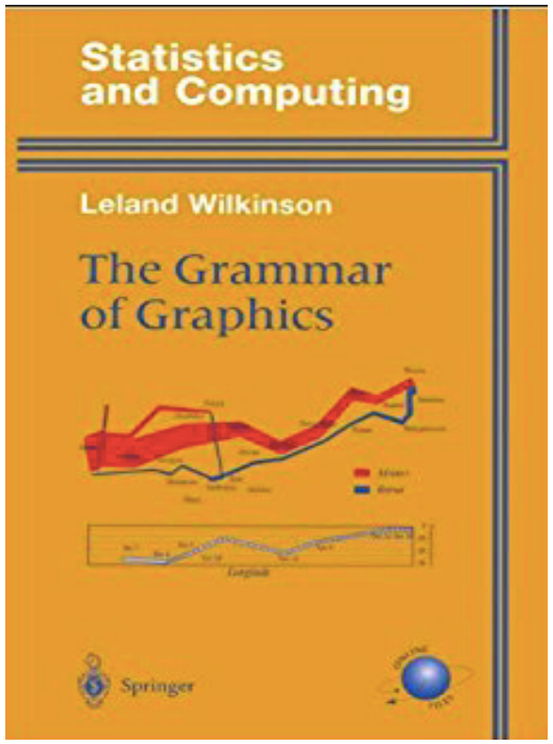
Thank you!

WELCOME TO GG PLOT2 Unit
ggplot2
PARENTAL
ADVISORY
EXPLICIT CONTENT
R Studio

Good grammar = how to build a plot in a logical way

Some history

- ggplot = grammar of graphics plot
  - Bertin (1983)
  - Based on a book called The Grammar of Graphics by Lenland Wilkinson (SYSTAT, SPSS, Tableau, H_2O)
  - “Good grammar” – what are the ingredients I need for a plot ?
    - Data
    - Axes
    - Stuff I do with the data (dots, lines, bars)
  - Inspired Hadley Wickham to develop ‘ggplot2’ package for R



Why ggplot2?

- What it needs to run:
  - A set of "geometries" – i.e. how we represent our data – scatter, bar, histogram etc
  - A set of "aesthetics" – i.e. things we do to make our data more informative – change size, shape etc. of dots, bars etc.

```
badfit %>%

  unnest(data) %>%

  ggplot(aes(x = year, y = lifeExp,
             colour = country)) +

  geom_line() +

  xlab("Year") +

  ylab("Life Expectancy in Years") +

  scale_colour_manual(
    name = "",
    values = cbbPalette[1:6] ) +

  ggtitle("Life Expectancy for \n
           Worst Fitting Models Over Time")
```

Things to be aware of (before you get your data)

- Short vs long format
- ggplot likes data frames tibbles

```
> head(data)
# A tibble: 6 x 11
      X subj_no   age   day  stim trial difficulty  resp  corr conf     RT
  <int>   <int> <int> <int> <int> <int>      <int> <int> <int> <fct> <dbl>
1     1      10    23     1     2     1          2     1     0 2     0.608
2     2      10    36     1     1     2          3     2     0 1     0.709
3     3      10    38     1     2     3          2     2     1 4     0.623
4     4      10    25     1     2     4          1     1     0 3     0.688
5     5      10    20     1     1     5          2     1     1 4     0.730
6     6      10    40     1     1     6          1     1     1 5     0.410
>
```

ggplot uses layers

1. The basics - layers

ggplot2 uses layers – keep adding until you like plot

data (cake)

asethetics - x, y, size, shape, colour (colours of icing, shape of cake)

geometries points, line, bar, text (layers of cake) – these also have aesthetics

statistics - regression line, binning, confidence intervals (garnish)

Let's start with the ingredients for the cake (load data)

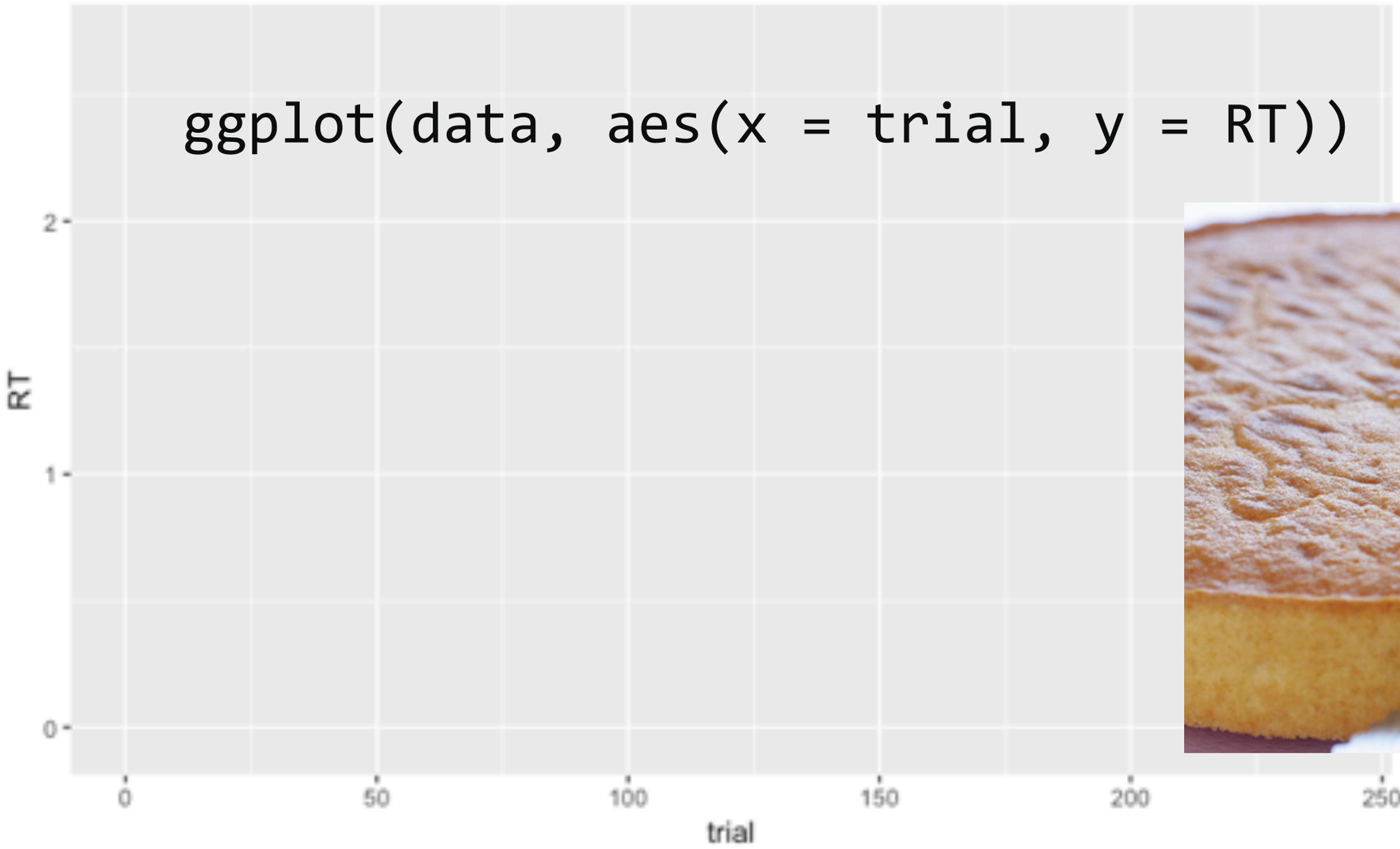
```{r create_plot_space}

# The first thing you need to do is tell ggplot what data you want to plot
ggplot(data)
```

This first layer is the data

- ggplot(data) tells ggplot2 you want to use the data frame called "data"
- It doesn't give you a plot (just grey box) because you haven't said what columns you want to plot yet!
  - Waiting for aesthetics - shape of cake, size of cake, colour, how many layers

Let's start with the data/cake (what kind of cake?)

```{r}

# Tell ggplot what bits of the data you want to plot
# ggplot data x = trial, y = RT
ggplot(data, aes(x = trial, y = RT))

```



Aesthetic to data mapping (aes)

statistics - regression line, binning, confidence intervals (garnish)

asethetics - x, y, size, shape, colour (colours of icing, shape of cake)

coordinate system(base)

data (cake)

Aesthetic to data mapping (aes)

```{r create_plot_space}

# The first thing you need to do is tell ggplot what data y
ggplot(data)

```

```{r}

# Step1 - Tell ggplot what bits of the data you want to plot ggplot data x = trial, y = RT_daty1
ggplot(data, aes(x = trial, y = RT_day1))


```


```{r geom_points_RT}
# Step 2 - Tell ggplot what you want to display it as points
ggplot(data, aes(x = trial, y = RT_day1)) + geom_point()

```

Aesthetic to data mapping (aes)

geometries points, line, bar, text (layers of cake) – these also have asthetics

asethetics - x, y, size, shape, colour (colours of icing, shape of cake)

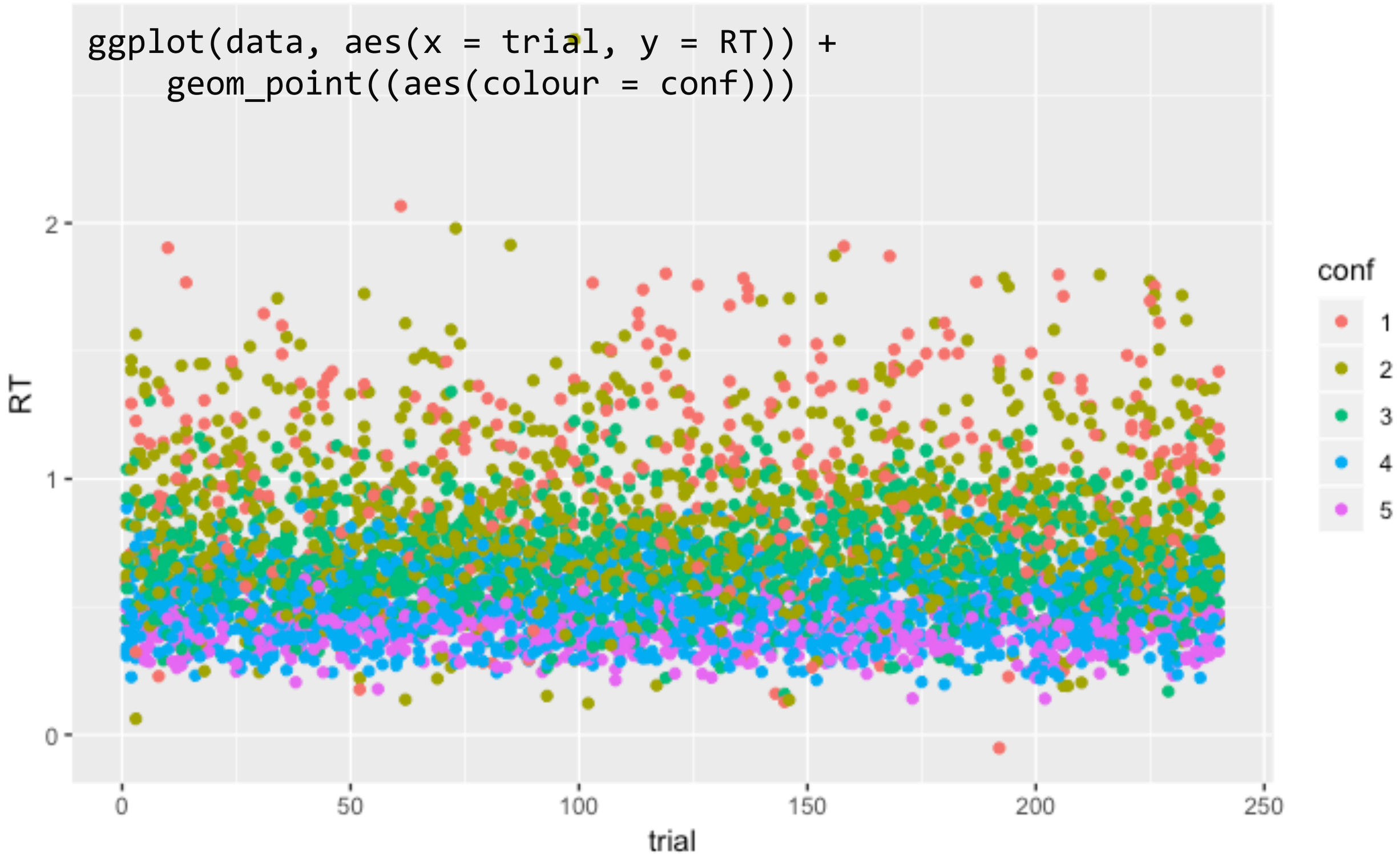
coordinate system(base)

data (cake)

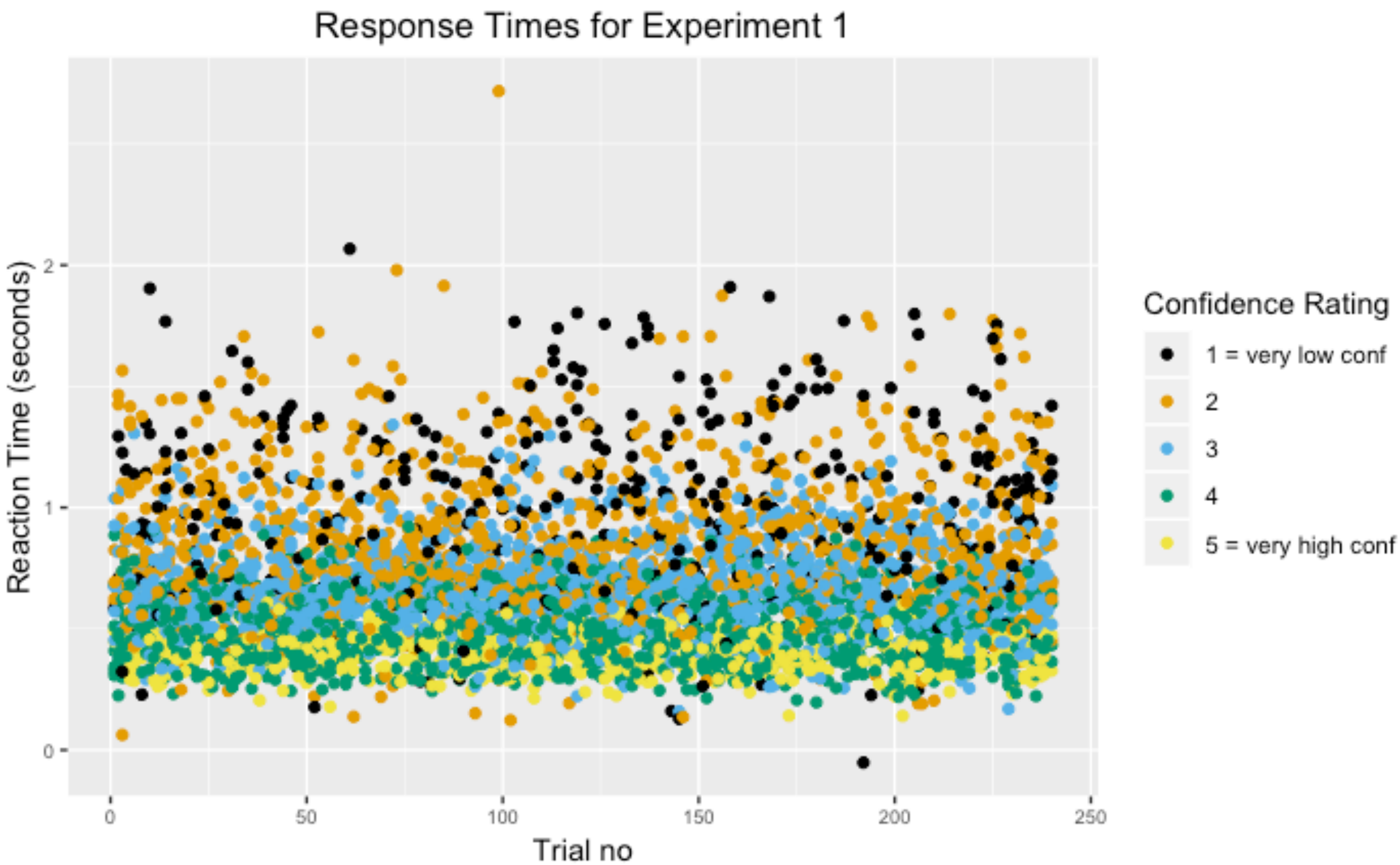
Aesthetic to data mapping (aes)

```
data$conf <- as.factor(data$conf) # convert data to a factor
```

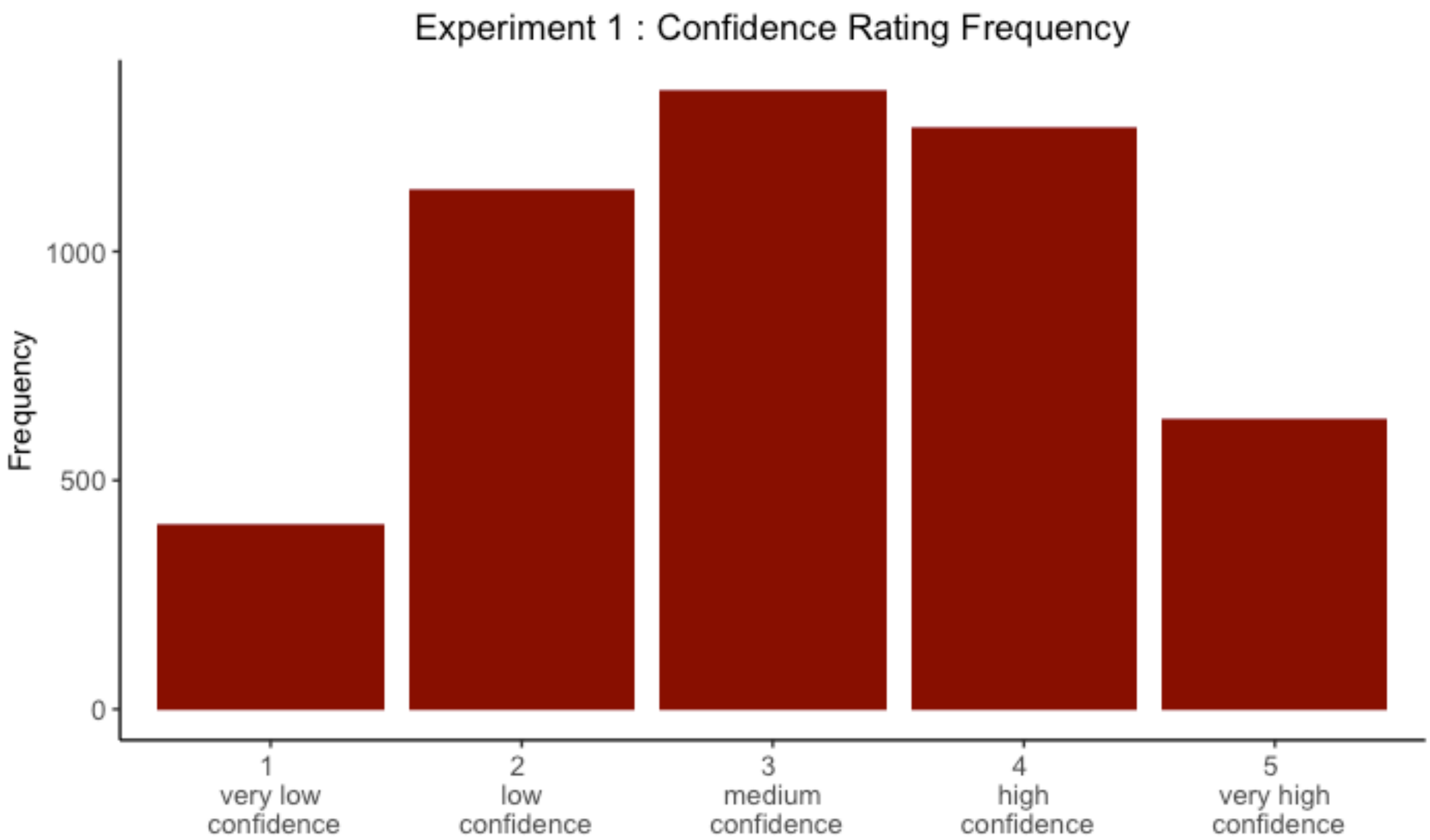
```
ggplot(data, aes(x = trial, y = RT)) +
    geom_point((aes(colour = conf)))
```



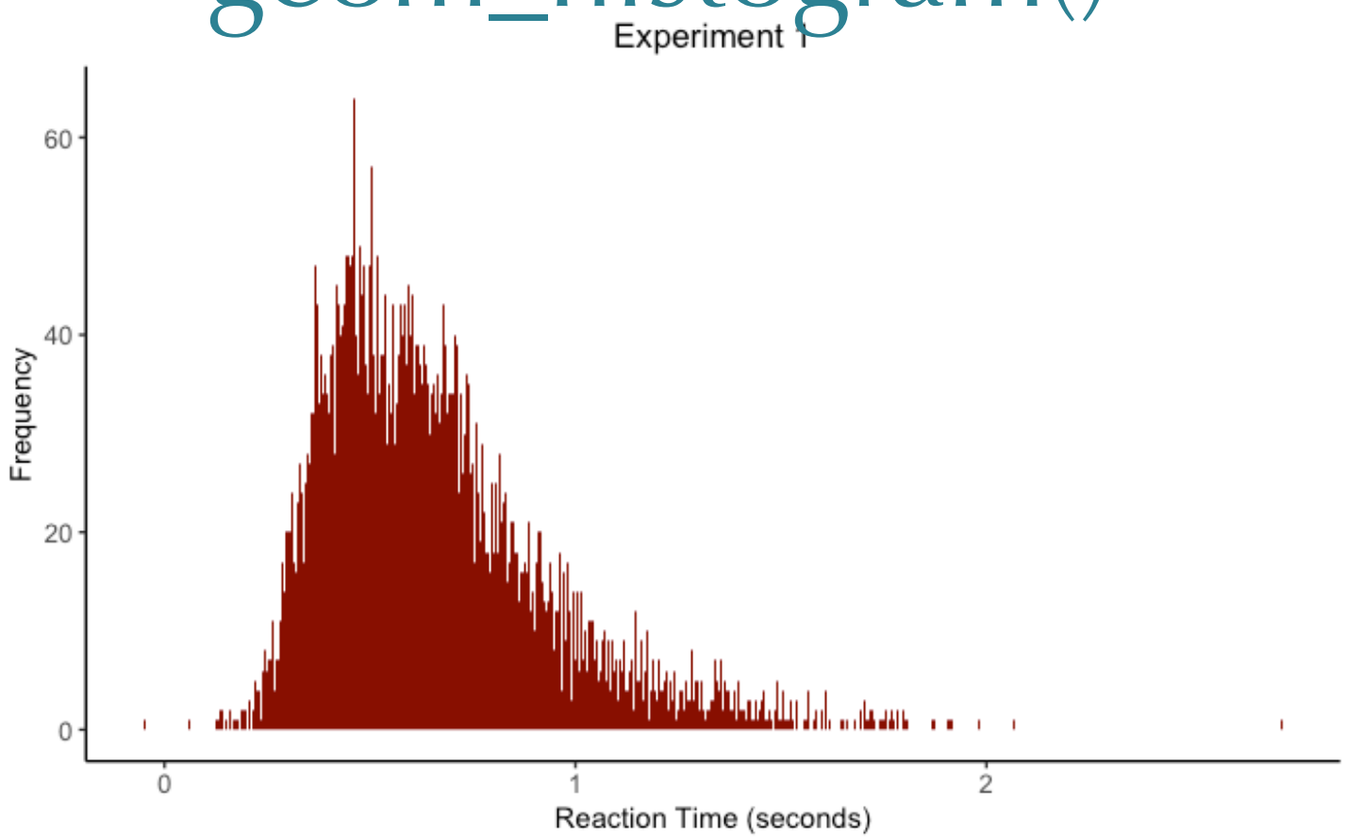
Colourblind palette



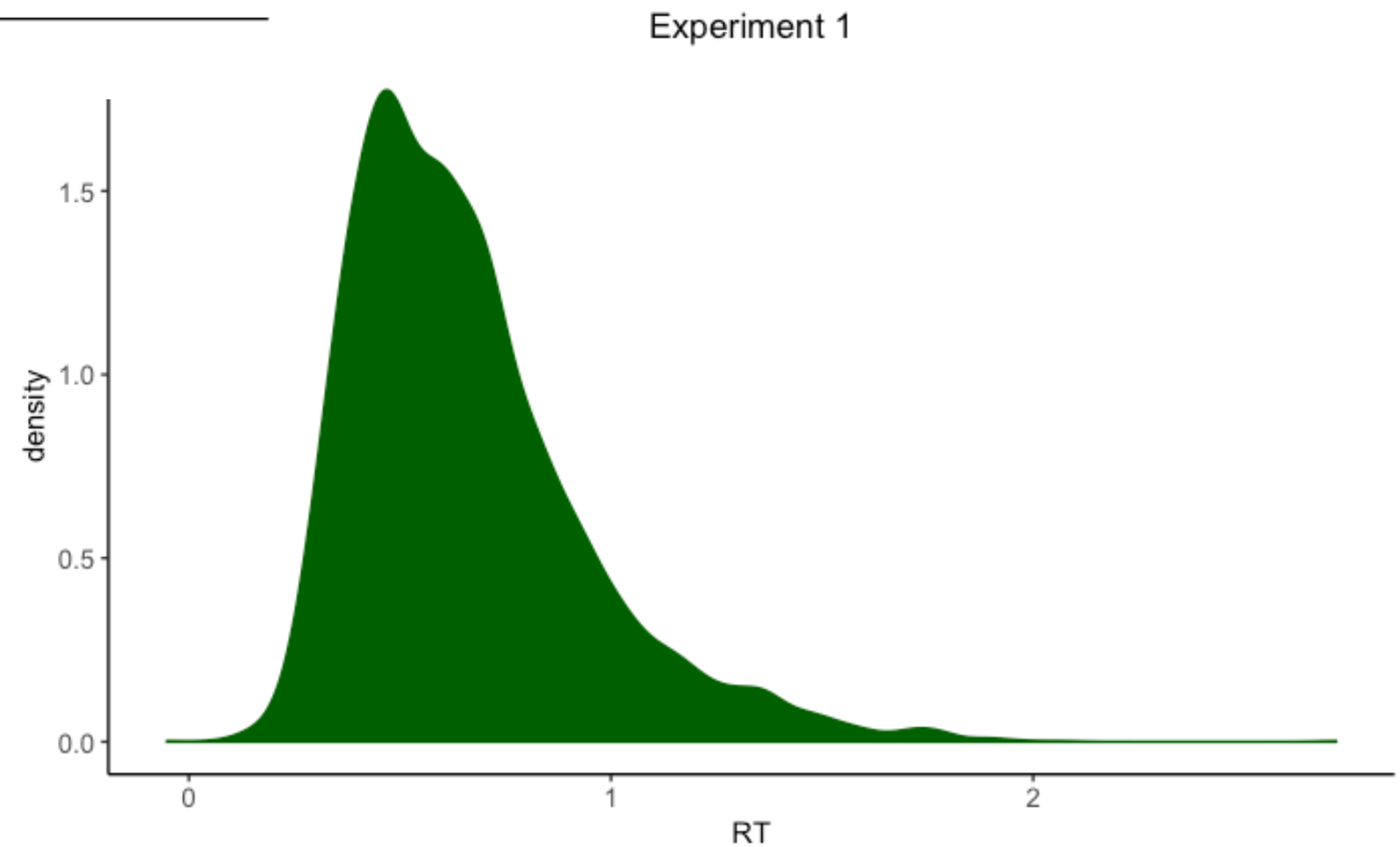
Barplots - geom_bar()



Histograms
geom_histogram()

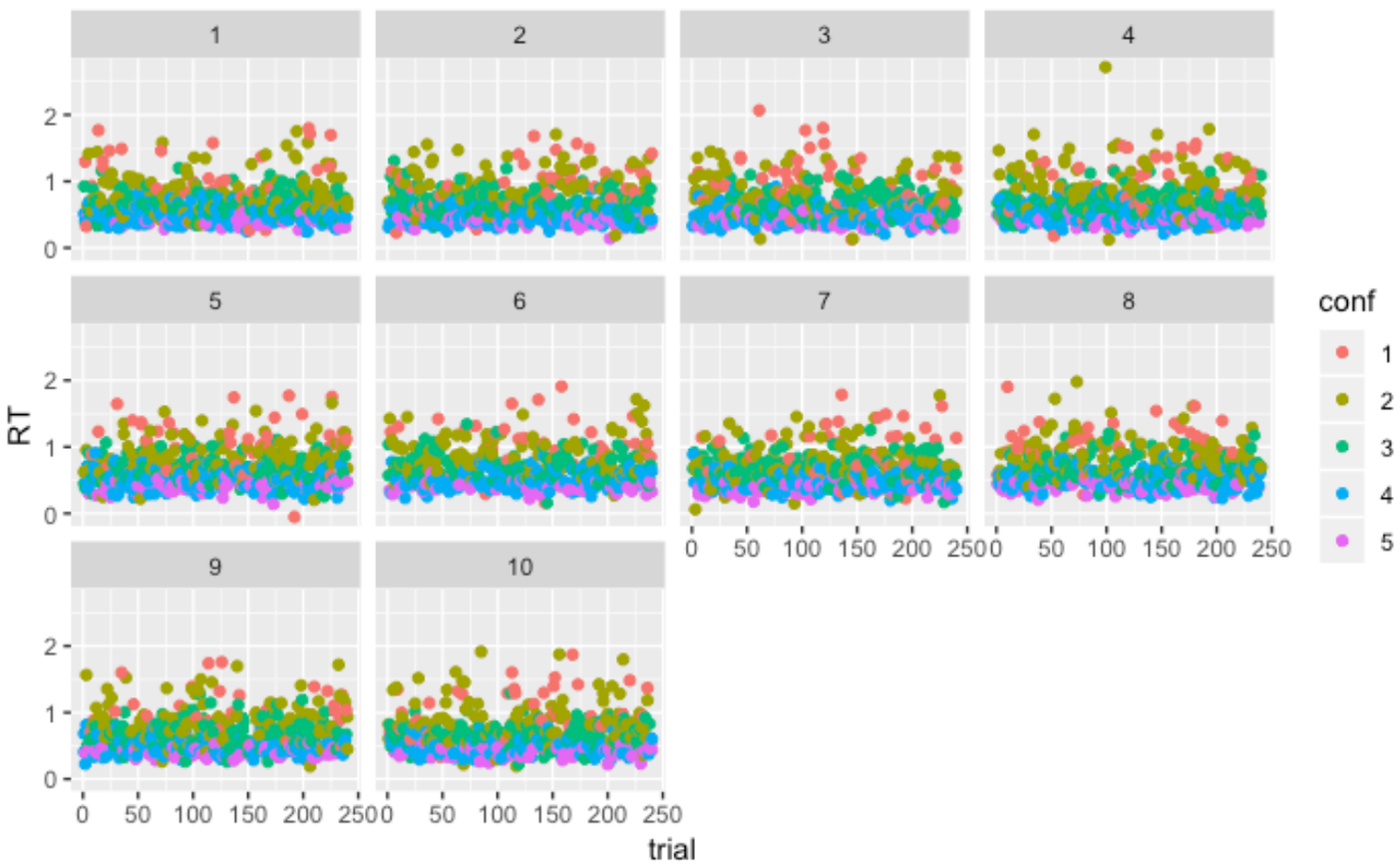


Density plots
geom_density()



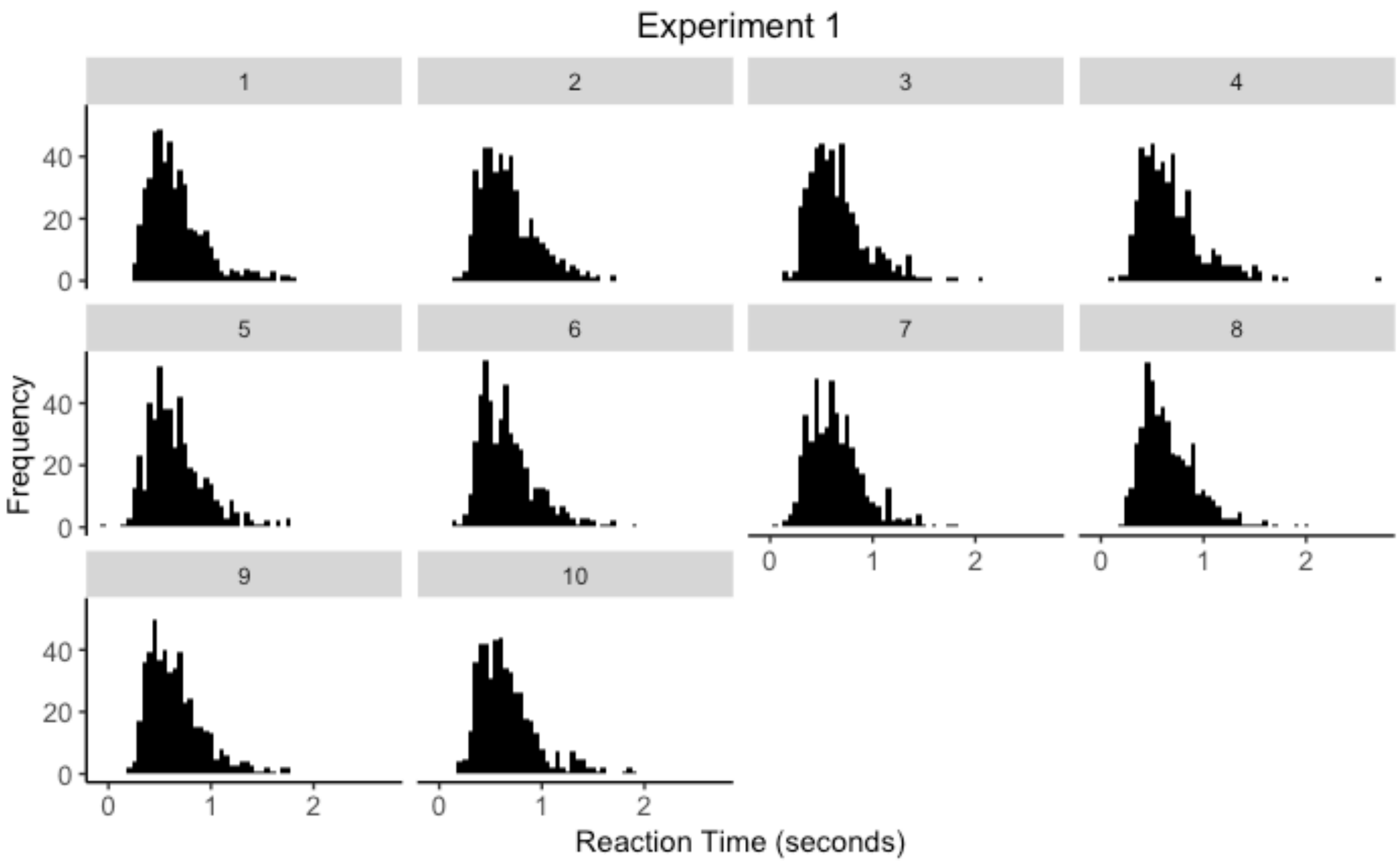
Plotting multiple subjects

```
ggplot(data, aes(x = trial, y = RT)) +
geom_point(aes(colour = conf)) +
facet_wrap(~subj_no)
```

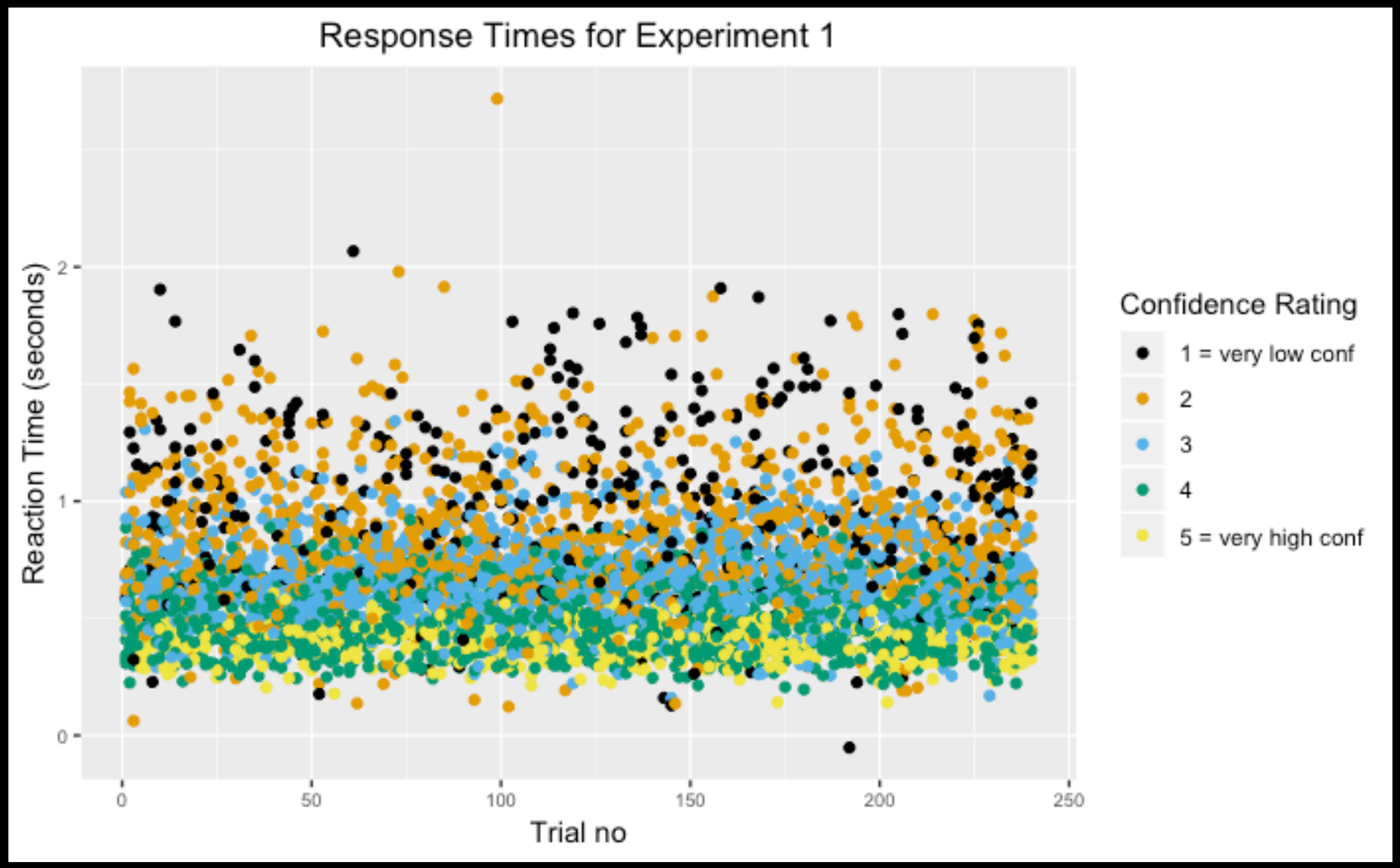


- Facet wrap is doing the work here
- facet_wrap(~column)
- Where subj_no is column in the data frame we are plotting

Plotting multiple subjects



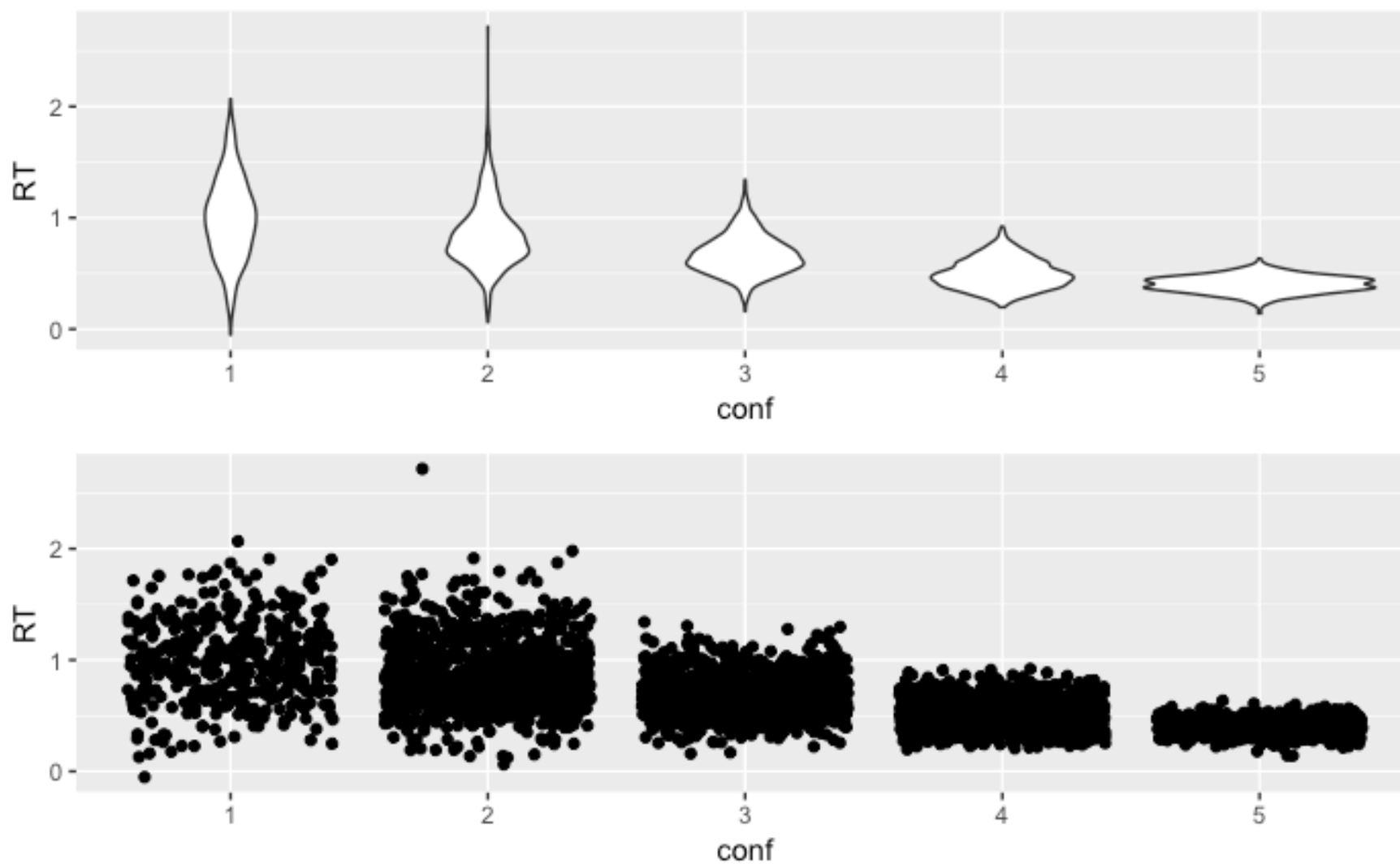
Colours and legends



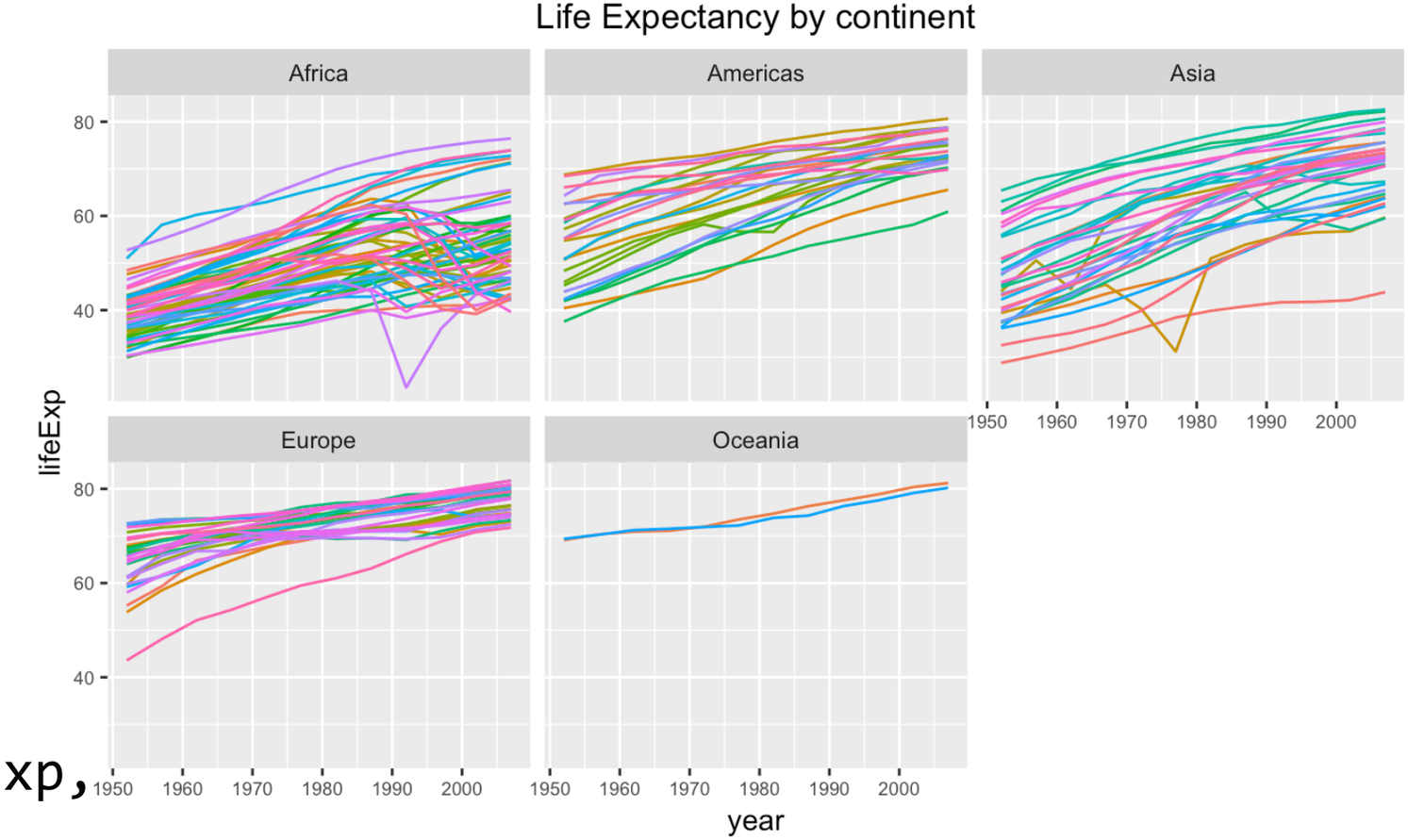
```
scale_color_manual(name = "Confidence Rating",
                   values = cbbPalette[1:5],
                   labels = c("1 = very low conf",
"2",
“3",
“4",
"5 = very high conf”)
```

Grid plotting

```
library(gridExtra)

violinPlot <- ggplot(data, aes(x = conf, y = RT)) + geom_violin()

jitterPlot <- ggplot(data, aes(y = RT, x = conf)) + geom_jitter()

jitterPlotfigure1 <- grid.arrange(violinPlot, jitterPlot,
             ncol = 1, nrow = 2)
```



Pipe -> ggplot2




```
gapminder.byContinent %>%

  unnest(data) %>%

  ggplot(., aes(x = year, y=lifeExp,
          group = country)) +

  geom_line(aes(color = country)) +

  facet_wrap(~continent) +

  ggtitle("Life Expectancy by continent") +

  theme(
    legend.position = "none",
    axis.text.x = element_text(size=7),  # size of text on the x axis
    axis.text.y = element_text(size=7),  # size of text on the y axis
    plot.title = element_text(hjust = 0.5, size = 13),  # title settings
    )
```

Pipe -> ggplot2 - video

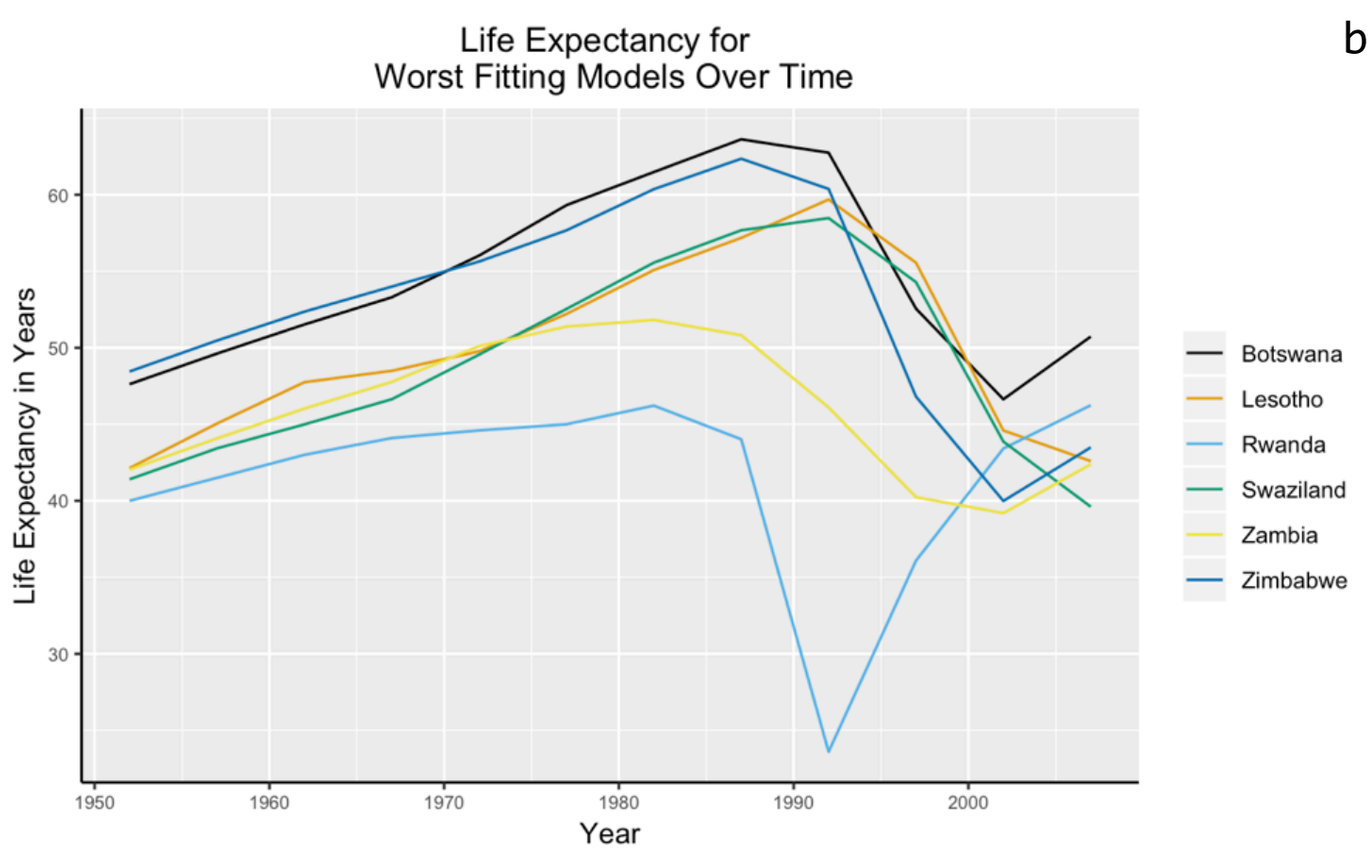
```
mutate(fit = map(data, ~lm(gdpPercap ~ lifeExp + pop, data = .)))

gapminder %>%
group_by(continent) %>%
nest() %>%
unnest(data) %>%
ggplot(aes(x = year, y=lifeExp, group = country)) +
geom_line(aes(color = country)) +
facet_wrap(~continent) +
ggtitle("Life Expectancy by continent") +
theme(
  legend.position = "none",
  axis.text.x = element_text(size=7),  # size of text on the x axis
  axis.text.y = element_text(size=7),  # size of text on the y axis
  plot.title = element_text(hjust = 0.5, size = 13),  # title settings (note, centering requires h =
)
```

Finally....

Get all of the worst fitting models and plot them

```
badfit <- gapminder2.byCountry %>%
  unnest(summary.stats) %>%
  filter(., r.squared < 0.25)
```

country <fctr>	continent <fctr>	data <list>	fit <list>	std.beta <list>	preds <list>	resids <list>	r.squared <dbl>
Rwanda	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<dbl [12]>	0.01715964
Swaziland	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<dbl [12]>	0.06821087
Zambia	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<dbl [12]>	0.05983644
Zimbabwe	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<dbl [12]>	0.05623196
Botswana	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<dbl [12]>	0.03402340
Lesotho	Africa	<tibble>	<S3: lm>	<data.frame [1 × 5]>	<tibble>	<dbl [12]>	0.08485635




```
badfit %>%

  unnest(data) %>%

  ggplot(aes(x = year, y = lifeExp,
         colour = country)) +

  geom_line() +

  xlab("Year") +

  ylab("Life Expectancy in Years") +

  scale_colour_manual(
    name = "",
    values = cbbPalette[1:6] ) +

  ggtitle("Life Expectancy for \n
          Worst Fitting Models Over Time")
```