Introduction to Exploratory Data Analysis (EDA)

Understanding your data with summaries, graphs, and transformations

Sean Davis

March 30, 2022

1. What is EDA?
2. Questions guide EDA
3. EDA employs visualation
4. A practical example of EDA.

Section 1

What is EDA?

What is Exploratory Data Analysis?

Exploratory Data Analysis, or EDA, is an important step in any Data Analysis or Data Science project.

EDA is the process of investigating the dataset: - discover patterns within and between variables - find anomalies and outliers - form hypotheses based on our understanding of the dataset

Getting started with EDA

Start by:

Define the toolkit

Multiple toolkits are available for data analysis. In our case, we will be using R, but others might use python, Spark, Julia, Perl, or others.

Access and load data

Accessing and loading data can sometimes be a challenge, but a good toolkit will provide solutions for common data formats and types.

EDA is an iterative process

1. Generate questions about your data.
2. Search for answers by:
   - Visualizing data
   - Summarizing data
   - Transforming data
3. Refine your questions, generate new questions, and then repeat from step

EDA is a mindset

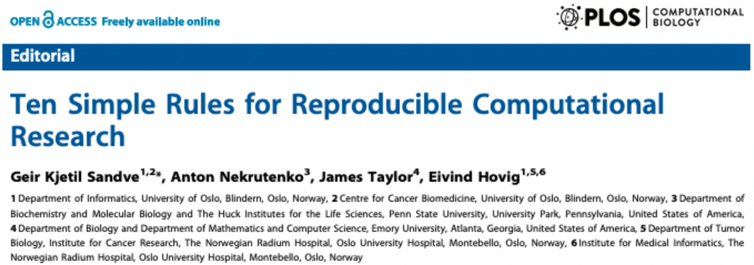
Figure 1: A growth mindset and curiosity are helpful when exploring data.

EDA is a mindset

- Exploratory data analysis is about playing with data.
- Curiosity and patience both play a part in successful EDA.
- There is not a set of rules for EDA.
- Collaboration and communication can add to the fun of EDA.
- As a data analysts or bioinformatician, sometimes EDA can lead to having to deliver bad news (failed experiment, lack of data to answer a question)

Reproducible research benefits from well-documented EDA

Reproducible computational research is a goal that we all aspire to [1].



OPEN ACCESS Freely available online

PLOS | COMPUTATIONAL BIOLOGY

Editorial

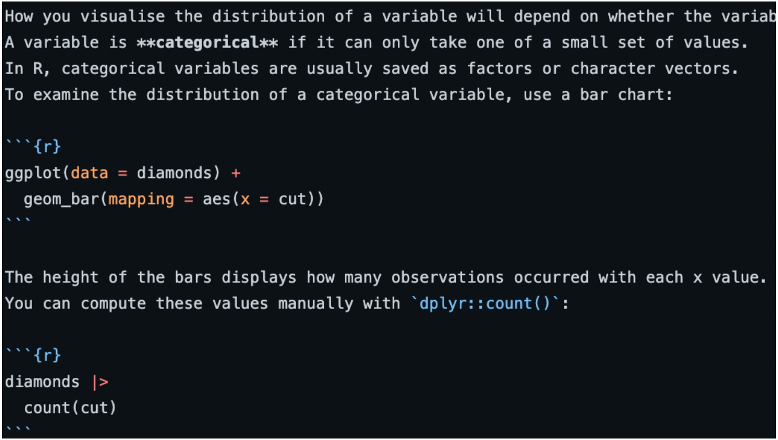
Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve[1,2*], Anton Nekrutenko[3], James Taylor[4], Eivind Hovig[1,5,6]

1 Department of Informatics, University of Oslo, Blindern, Oslo, Norway, 2 Centre for Cancer Biomedicine, University of Oslo, Blindern, Oslo, Norway, 3 Department of Biochemistry and Molecular Biology and The Huck Institutes for the Life Sciences, Penn State University, University Park, Pennsylvania, United States of America, 4 Department of Biology and Department of Mathematics and Computer Science, Emory University, Atlanta, Georgia, United States of America, 5 Department of Tumor Biology, Institute for Cancer Research, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, Oslo, Norway, 6 Institute for Medical Informatics, The Norwegian Radium Hospital, Oslo University Hospital, Montebello, Oslo, Norway

While perhaps a bit beyond the scope of this lecture, your future self will thank you if you carefully document your EDA to aid in reproducibility and reuse. R markdown is a great way to accomplish this.

Approach EDA as a lab notebook for data



~~~
How you visualise the distribution of a variable will depend on whether the variab
A variable is **categorical** if it can only take one of a small set of values.
In R, categorical variables are usually saved as factors or character vectors.
To examine the distribution of a categorical variable, use a bar chart:

```{r}
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut))
~~~

The height of the bars displays how many observations occurred with each x value.
You can compute these values manually with `dplyr::count()`:

```{r}
diamonds |>
  count(cut)
```
```

Figure 2: Use R markdown as your data science lab notebook.
```

Section 2

# Questions guide EDA

# Guide EDA with questions

*“There are no routine statistical questions, only questionable statistical routines.” — Sir David Cox*

*“Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise.” — John Tukey*

# Ask lots of questions of data

- A key to understanding data and generating new insight from them is to ask **lots** of questions.
- Document your questions and their answers, including the *how and* the *why*.
- Use the answer to previous questions to generate new ones.
- Some questions may be in the form of a *hypothesis* to be tested, but many will not.

# Answer two key questions to ask in EDA

**What is the *variation* in each of my variables, individually?**

Every variable has its own pattern of variation, which can reveal interesting information including quality issues like outliers.

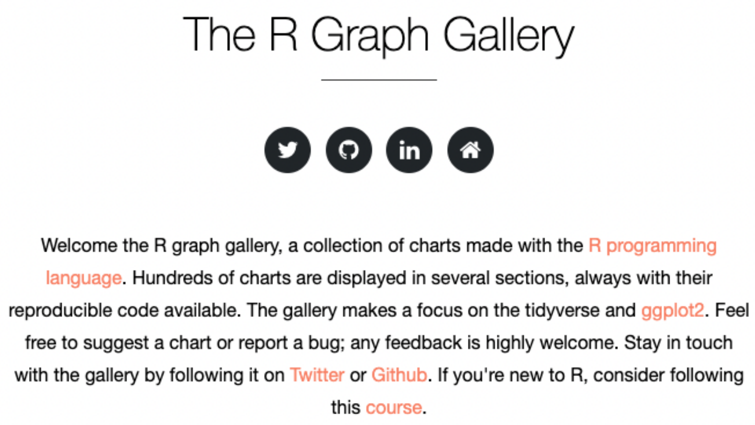
**What is the *covariation* between my variables?**

Covariation is the tendency for the values of two or more variables to vary together in a related way.

Section 3

# EDA employs visualation

# Visualization is a key component of EDA



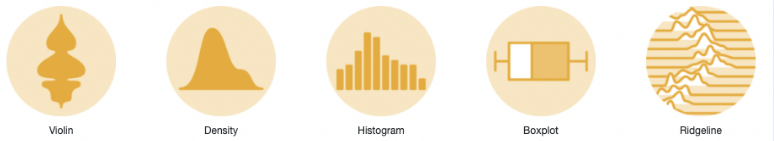
## The R Graph Gallery

Welcome the R graph gallery, a collection of charts made with the R programming language. Hundreds of charts are displayed in several sections, always with their reproducible code available. The gallery makes a focus on the tidyverse and ggplot2. Feel free to suggest a chart or report a bug; any feedback is highly welcome. Stay in touch with the gallery by following it on Twitter or Github. If you're new to R, consider following this course.

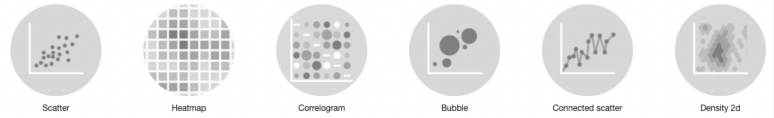
See The R Graph Gallery for an interactive web gallery of approaches to graphing data.

# Graphs can help answer questions about data.

## Distributions of a single variable

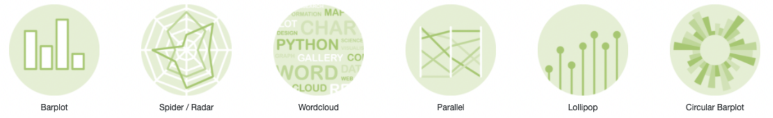


## Showing relationships between variables

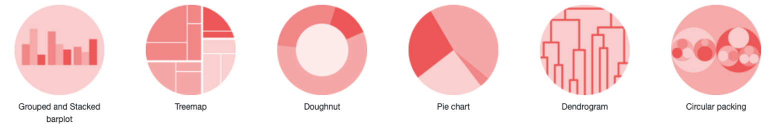


# Choosing the right graph conveys a story about the data

## Showing rankings or proportions

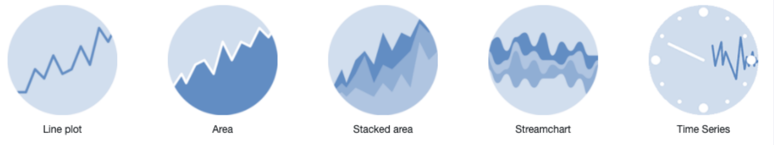


## Parts of a whole

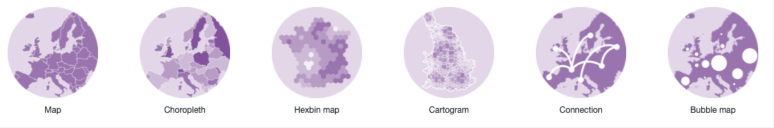


# Some graphs are very specific

## Time-ordered data



## Maps and spatial data



Section 4

# A practical example of EDA.

# Pick a dataset that interests you

We'll be working with the dataset described here: https://ggplot2.tidyverse.org/reference/mpg.html. Since this is a dataset that comes with the `ggplot2` package, you could also use this code to get details:

```
library(ggplot2)
help('mpg')
```

Dataset description

This dataset contains a subset of the fuel economy data that the EPA makes available on https://fueleconomy.gov/. It contains only models which had a new release every year between 1999 and 2008 - this was used as a proxy for the popularity of the car.

Load data and start exploring

```
library(ggplot2)
data(mpg)
```

What are the variable names?

```
colnames(mpg)
```

```
##  [1] "manufacturer" "model"        "displ"
##  [4] "year"         "cyl"          "trans"
##  [7] "drv"          "cty"          "hwy"
## [10] "fl"           "class"
```

How big are the data?

```
dim(mpg)
```

```
## [1] 234  11
```

What are the types of data in mpg?

```
sapply(mpg, class)
```

```
## manufacturer        model        displ
##  "character"  "character"    "numeric"
##         year          cyl        trans
##    "integer"    "integer"  "character"
##          drv          cty          hwy
##  "character"    "integer"    "integer"
##           fl        class
##  "character"  "character"
```

We can quickly summarize the data in mpg

```
summary(mpg)
```

```
##  manufacturer          model
##  Length:234         Length:234
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##      displ            year           cyl
##  Min.   :1.600   Min.   :1999   Min.   :4.000
##  1st Qu.:2.400   1st Qu.:1999   1st Qu.:4.000
##  Median :3.300   Median :2004   Median :6.000
##  Mean   :3.472   Mean   :2004   Mean   :5.889
##  3rd Qu.:4.600   3rd Qu.:2008   3rd Qu.:8.000
##  Max.   :7.000   Max.   :2008   Max.   :8.000
##     trans               drv
```

We can get a glimpse of the values in mpg

```
library(tidyverse)
glimpse(mpg)
```

```
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "au~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4~
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.~
## $ year         <int> 1999, 1999, 2008, 2008, 199~
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, ~
## $ trans        <chr> "auto(l5)", "manual(m5)", "~
## $ drv          <chr> "f", "f", "f", "f", "f", "f~
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18,~
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27,~
## $ fl           <chr> "p", "p", "p", "p", "p", "p~
## $ class        <chr> "compact", "compact", "comp~
```

The manufacturer variable is categorical

```
unique(mpg$manufacturer)
```

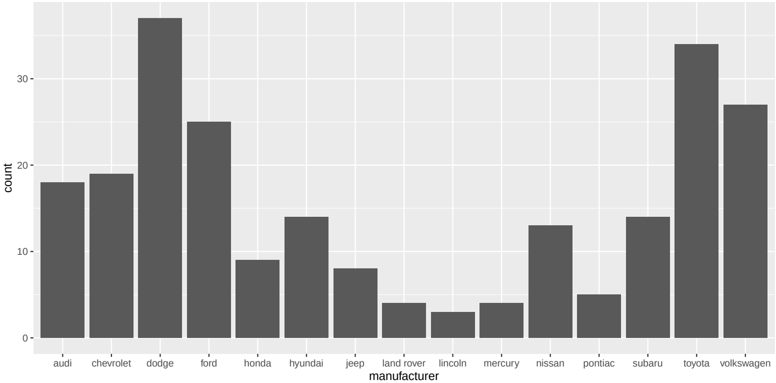
```
##  [1] "audi"         "chevrolet"    "dodge"
##  [4] "ford"         "honda"        "hyundai"
##  [7] "jeep"         "land rover"   "lincoln"
## [10] "mercury"      "nissan"       "pontiac"
## [13] "subaru"       "toyota"       "volkswagen"
```

```
table(mpg$manufacturer)
```

```
##
##        audi  chevrolet      dodge       ford
##          18         19         37         25
##       honda    hyundai       jeep land rover
##           9         14          8          4
##     lincoln    mercury     nissan    pontiac
##           3          4         13          5
```

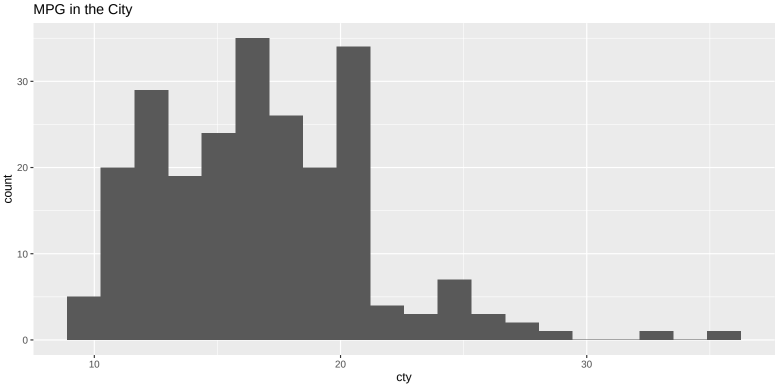
We can visualize categorical variable distribution using barplots

```
ggplot(mpg,mapping = aes(x = manufacturer)) +
  geom_bar()
```



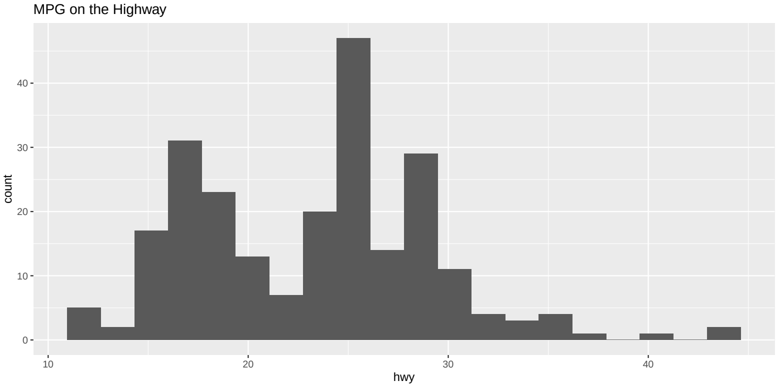
We can visualize continuous variables using histograms

```
ggplot(mpg, mapping = aes(x = cty)) +
  geom_histogram(bins=20) + ggtitle('MPG in the City'
```



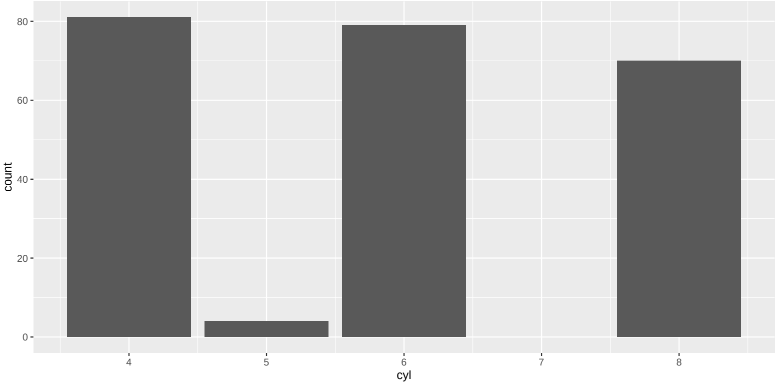
We can visualize continuous variables using histograms

```
ggplot(mpg, mapping = aes(x = hwy)) +
  geom_histogram(bins=20) + ggtitle('MPG on the Highw
```



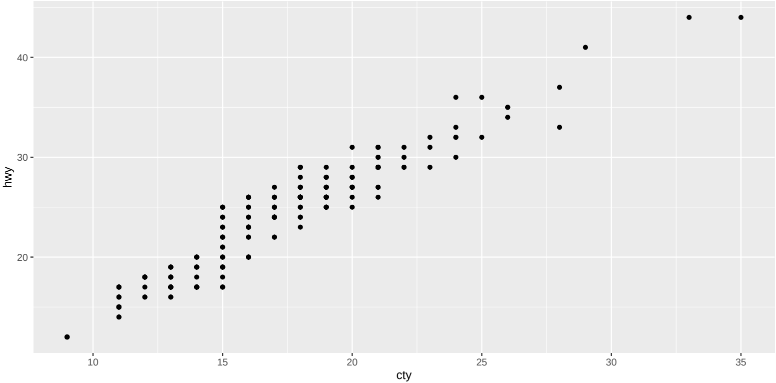
Some numeric variables are also categorical

```
ggplot(mpg, mapping = aes(x = cyl)) +
  geom_bar()
```



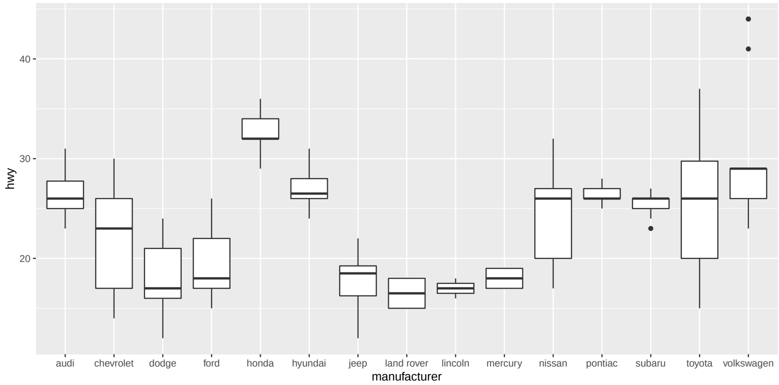
Use a scatterplot to relate two numeric variables

```
ggplot(mpg, mapping = aes(x = cty, y=hwy)) +
  geom_point()
```



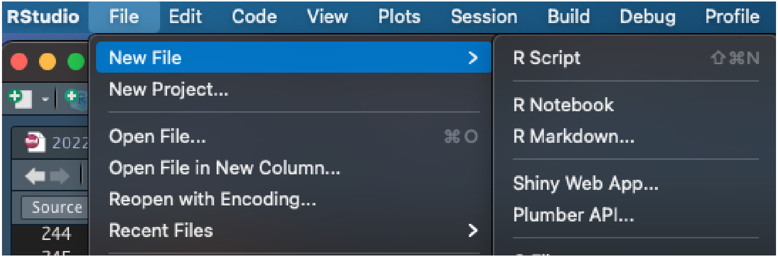
Use a boxplot for a categorical and numeric variable

```
ggplot(mpg, mapping = aes(x = manufacturer, y=hwy)) +
  geom_boxplot()
```



# Using Rmarkdown

Start with a blank Rmarkdown



Use Rmarkdown headers to organize your thoughts

- Introduction and background
- Dataset(s)
  - include lots of descriptive plots and tables
- Results
  - ask and answer questions here
- Conclusions (can also go with the questions and answers)
- Future work and extensions
  - Document questions that you think you'd like to answer later, including why.
- Use headers for questions

Additional Rmarkdown tips

- Use R blocks to write R code
- `knit your rmarkdown` regularly to check for errors and results
- Use the R console to try and perfect code and then add to the Rmarkdown document
- Don't forget to explain in text your rationale for asking a question of your data
- Don't forget to write down your explanation of your findings, knowing that your *future self* is a key reader

[1] Geir Kjetil Sandve et al. "Ten simple rules for reproducible computational research". In: *PLoS computational biology* 9.10 (Oct. 2013), e1003285. ISSN: 1553-734X, 1553-7358. DOI: 10.1371/journal.pcbi.1003285. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3812051&tool=pmcentrez&rendertype=abstract.