Introduction to data analysis

Part 6. EDA in R

Let's get started.

the master labs • academy

NCOI LEARNING

Table of Contents

In this chapter you will learn:

- Why we explore data
- The types of insights we can gather

It's all about **gaining insight**

- Understand what has happened or will happen
- Understand what is happening
- Predict what will happen
- Help us make decisions

It's all about **gaining insight**

*The art of looking at one or more datasets in an effort to understand **the underlying structure** of the data contained there.*

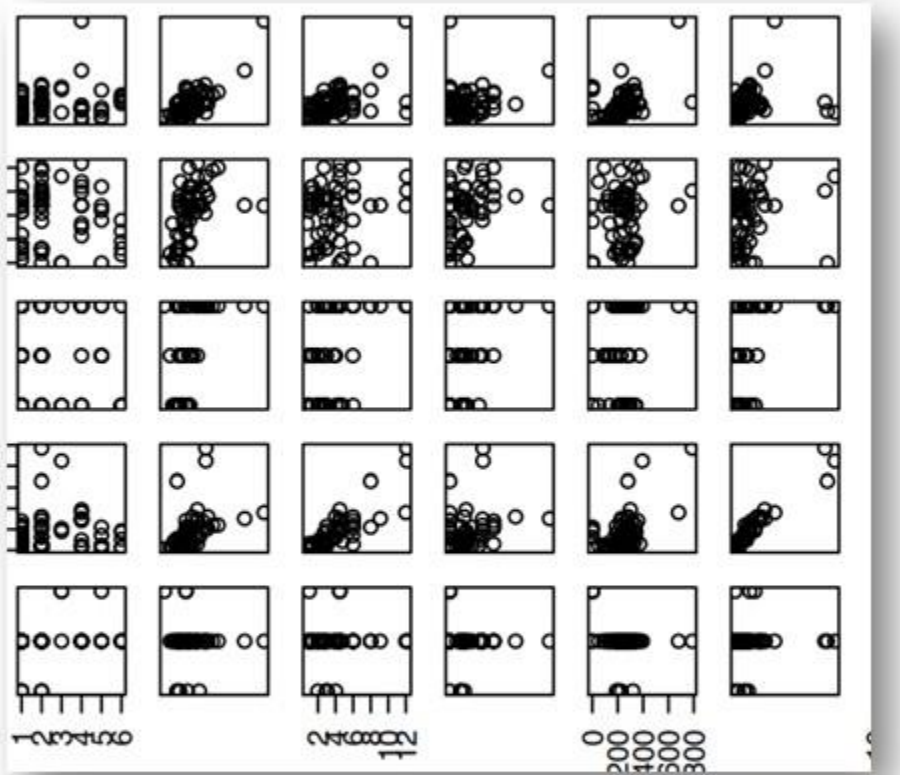
It's all about
gaining insight

... by asking questions about the data.

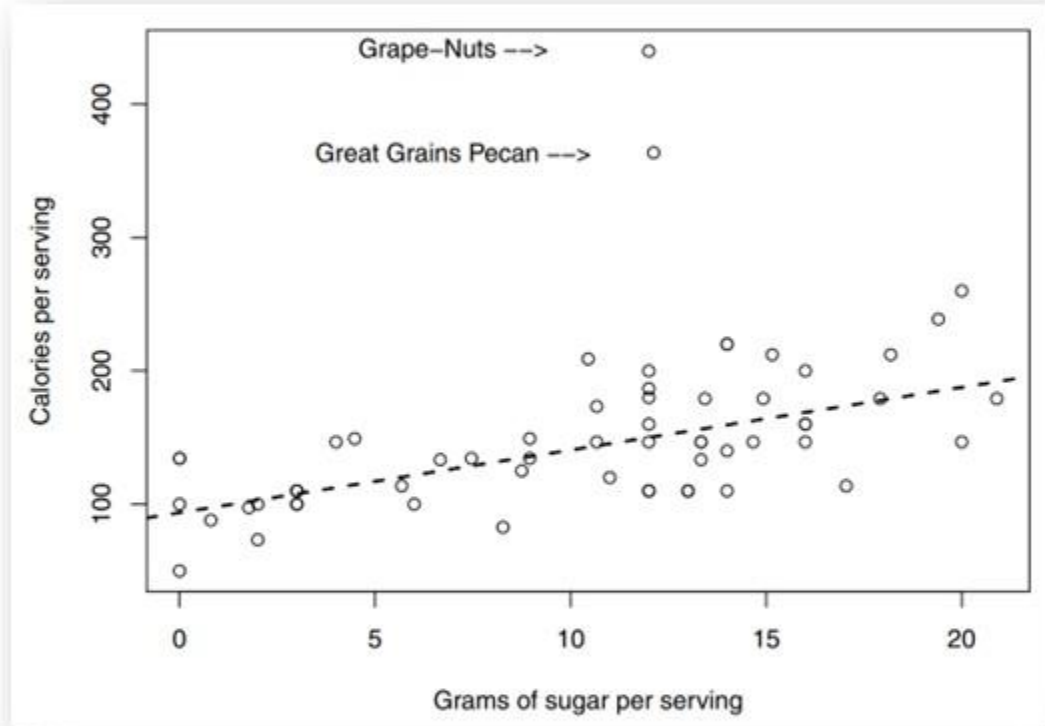
- Amount of data
- Data types
- Outliers
- Missing values
- Distributions
- Connections between data

Exploratory vs explanatory

data analysis



Get a sense of what's inside



Convey conclusions to others

Table of Contents

In this chapter you will learn:

- The steps to take to perform exploratory analysis in R
- The keywords that represent each step

A general **strategy**

1. Assess the basic characteristics of the dataset

Keywords:

- Record amount
- Datatypes
- Variable names
- Outliers
- Missing values

A general **strategy**

1. Assess the basic characteristics of the dataset

2. Examine descriptive statistics for each variable

Keywords:

- Mean, mode, median
- Max, min
- Standard deviation
- Range

A general **strategy**

1. Assess the basic characteristics of the dataset
2. Examine descriptive statistics for each variable
3. Where possible or valuable examine exploratory visualizations

Keywords:

- Ggplot
- Barchart
- Boxplot
- Histogram
- ...

A general **strategy**

1. Assess the basic characteristics of the dataset
2. Examine descriptive statistics for each variable
3. Where possible or valuable examine exploratory visualizations
4. Where possible, apply the procedures to look for data anomalies

Keywords:

- QQ-plot
- Boxplot
- Scatterplot

A general **strategy**

1. Assess the basic characteristics of the dataset
2. Examine descriptive statistics for each variable
3. Where possible or valuable examine exploratory visualizations
4. Where possible, apply the procedures to look for data anomalies
5. Look at the relations between key variables

Keywords:

- Scatterplot
- Boxplot
- Mosaic plot

Table of Contents

In this chapter you will learn:

- How to read data with R
- The different types of data
- How to determine the basic aspects of the data

Examining basic data characteristics

Questions to ask

How many records do we have? How many variables?

What are the variable names? Are they meaningful?

What type is each variable—e.g., numeric, categorical, logical?

How many unique values does each variable have?

What value occurs most frequently, and how often does it occur?

Are there missing observations? If so, how frequently does this occur?

BUT FIRST...

Examining basic data characteristics

Requirements

Steps

- A way to load the data
- A way to look at the data
  - Data types
  - Possible values

Ingredients

- TidyVerse
- Readr
  - Glimpse
  - Df_status
  - Freq
  - Describe

Examining basic data characteristics

About tidyverse

readr

Use to read data

Needs:
- File type
- Read parameters

dplyr

Use to manipulate data

Needs:
- Data
- Type of manipulation

ggplot2

Use to create graphics

Needs:
- Data
- Aesthetic mapping

tidyr

Use to tidy up data

Needs:
- Data
- How to clean

Out of scope

About tidyverse

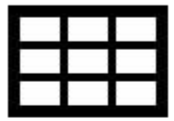
dplyr

Use to manipulate data

Needs:
- Data
- Type of manipulation

```
glimpse(x, width = NULL, ...)
```

Examining basic data characteristics:
Loading the data



A .txt or a tab-delimited text file can be read with the basic R function read.table()

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/read.table

Examining basic data characteristics:
Loading the data

A .csv or a comma-separated file (or semicolon) can be read with the R function read.csv() or read.csv2()

```
read.csv(file, header = FALSE, sep = "", quote = "\"'",
         dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
         row.names, col.names, as.is = !stringsAsFactors,
         na.strings = "NA", colClasses = NA, nrows = -1,
         skip = 0, check.names = TRUE, fill = !blank.lines.skip,
         strip.white = FALSE, blank.lines.skip = TRUE,
         comment.char = "#",
         allowEscapes = FALSE, flush = FALSE,
         stringsAsFactors = default.stringsAsFactors(),
         fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/read.table

Examining basic data characteristics:
Loading the data

Makes reading tab-delimited data a bit easier by choosing certain default parameters.

```
read.delim(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/read.table

Examining basic data characteristics:
Loading the data

Other options include:

read_excel (requires readxl package)

fromJSON (requires rjson package)

xmlTreeParse (requires XML package)

Examining basic data characteristics:
Loading the data

Questions to ask

How many records do we have? How many variables?

Retrieves the dimension of an object

```
dim(x)
```

```
```{r}
dim(oil)
```
```

```
[1] 84  2
```

Questions to ask

How many records do we have? How many variables?

Examining basic data characteristics

Questions to ask

How many records do we have? How many variables?

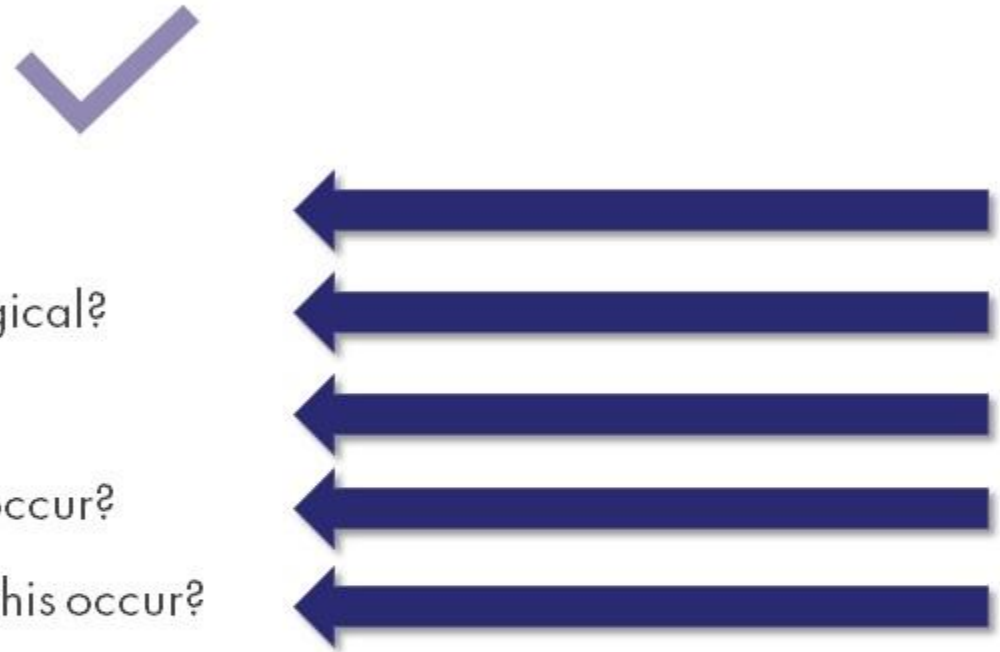
What are the variable names? Are they meaningful?

What type is each variable—e.g., numeric, categorical, logical?

How many unique values does each variable have?

What value occurs most frequently, and how often does it occur?

Are there missing observations? If so, how frequently does this occur?



Questions to ask

What are the variable names? What are their types? ...

summary (requires dlookr package)

```
summary(object, …)
```

str

```
str(object, …)
```

Examining basic data characteristics

Questions to ask

Other options include:

describe (requires dlookr package)

```
describe(.data, ...)
```

df_status (requires funModeling package)

```
df_status(data, print_results)
```

Questions to ask

What are the variable names?

What are their types?

How many unique values?

Which are the most frequent?

Any missing values?

Examining basic data characteristics

Questions to ask

What is the range of stores?

Does the test data contain NA-values?

What is the standard deviation of the train data unit_sales column?

Which variables does the oil file contain?

It's all about **gaining insight**

Data types

Categorical

Categories or groups

Numerical

Well... numbers

It's all about **gaining insight**

Data types

Discrete

Can only take on a certain number of values

Continuous

Can take on an infinite number of possibilities

It's all about **gaining insight**

Data types

Discrete

Can only take on a certain number of values

Number of people passing the statistics course

Number of people attending a concert, resulting in buying a CD

Continuous

Can take on an infinite number of possibilities

Length Time Weight

It's all about **gaining insight**

Data types

Binary

0 - 1

Yes - No

groups: 2
Sequence: No

Nominal

Group 3

Group 1

Group 2

groups: >2
Sequence: No

Ordinal

Runner 1

Runner 2

Runner 3

groups: >2
Sequence: Yes

It's all about **gaining insight**

Data types

Interval

°C

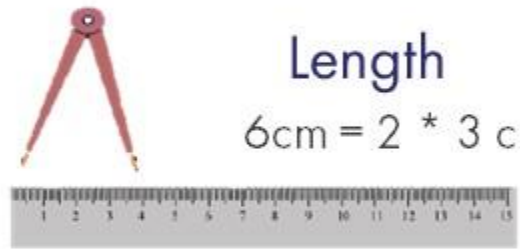
20° is not 4 * 5°

Clock time

4 p.m. is not twice as late as 2 p.m.

Natural origin: No
Meaningful ratios: No

Ratio



Length

6cm = 2 * 3 cm

°Kelvin

Natural origin: Yes
Meaningful ratios: Yes

Table of Contents

In this chapter you will learn:

- The different options for descriptive statistics based on data type
- How to take a first look at the actual data

Descriptive statistics for categorical data

Questions to ask

How many levels does the variable have?

How often do these levels occur?

How does one variable X vary of the levels?

… so we need to measure anyway

Descriptive statistics for numerical data

	Discrete		Continuous
	Nominal	Ordinal	Interval/ratio
Central tendency	Mode	Mode Median	Mode Median Arithmetic mean Geometric mean
Variability		Range Quartiles Interquartile range	Range Quartiles Interquartile range Variance Standard deviation Coefficient of variance
Central tendency and skewness		Boxplot	Boxplot
Correlation			Correlation coefficient Covariance

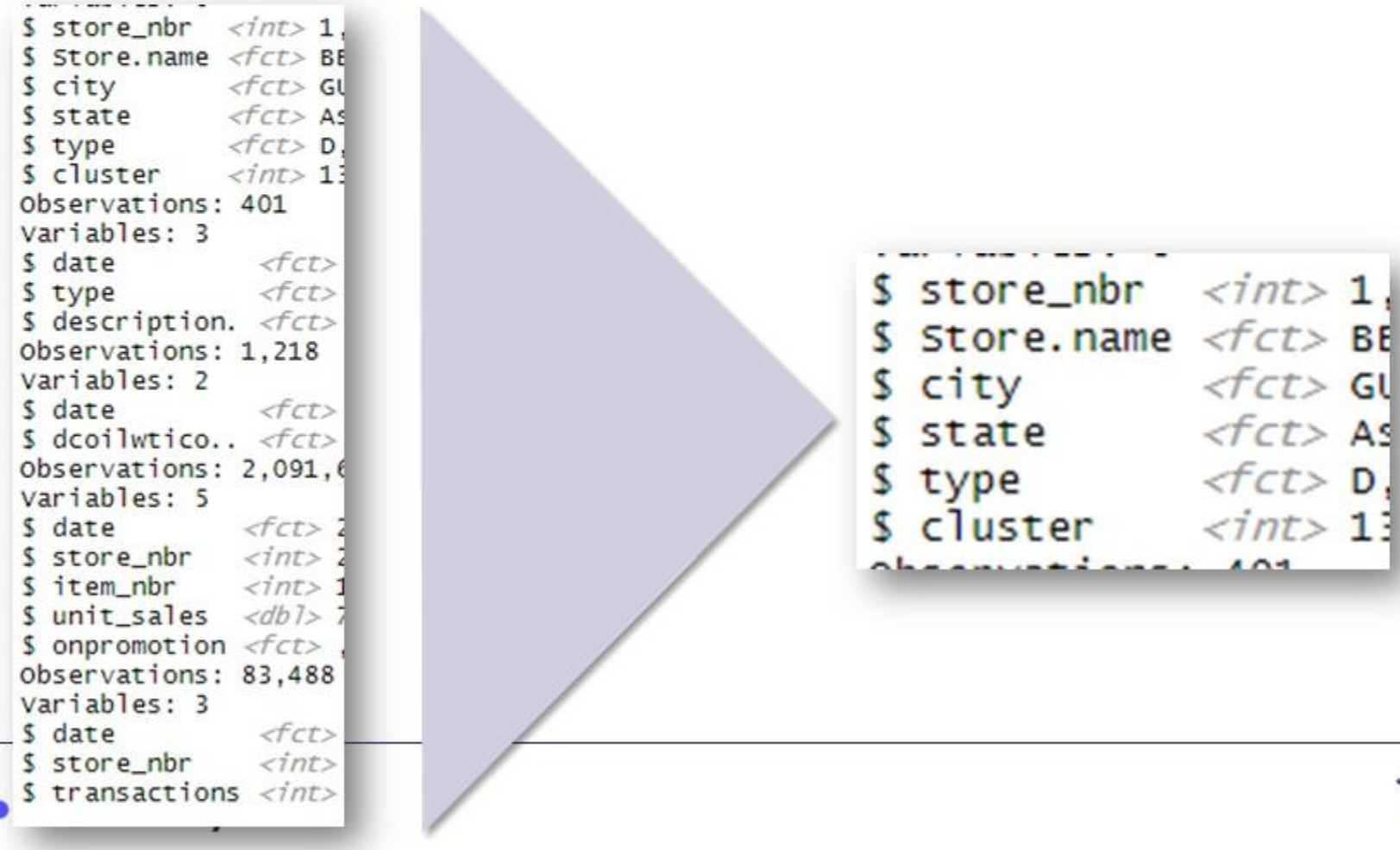
Checking out the categorical data

Make a list of all categorical data elements in the provided files. (HINT: use the glimpse function and look for factors)

```
Observations: 54
Variables: 6
$ store_nbr   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, ...
$ Store.name  <fct> BB - FC-GUWAHATI-BHANAGAGHAR-CITY SQUARE  G.S ROAD, BB-JOHRAT-MOUZA- NAGARMHAL-PMS / B&A COMPLEX , BB-SILCHAR-PARGANA-BARAK...
$ city        <fct> GUWAHATI, JOHRAT, SILCHAR, GUWAHATI, PATNA, PATNA, BHAGALPUR, PATNA, RAIPUR, BILASPUR, BHILAI, RAIPUR, DHANBAD, RANCHI, JAM...
$ state       <fct> Assam, Assam, Assam, Assam, Bihar, Bihar, Bihar, Bihar, Chhattisgarh, Chhattisgarh, Chhattisgarh, Chhattisgarh, Jharkhand, ...
$ type        <fct> D, D, D, D, D, D, D, D, B, C, B, C, C, C, C, C, C, B, C, B, B, C, D, D, D, D, D, E, E, C, B, C, C, B, C, E, D, D, B, C, D, ...
$ cluster     <int> 13, 13, 8, 9, 4, 13, 8, 8, 6, 15, 6, 15, 15, 7, 15, 3, 12, 16, 15, 6, 6, 7, 9, 1, 1, 10, 1, 10, 10, 3, 10, 3, 3, 6, 3, 10, ...
Observations: 401
Variables: 3
$ date          <fct> 2012-01-01, 2012-01-05, 2012-01-14, 2012-01-15, 2012-01-05, 2012-01-26, 2012-01-28, 2012-02-5, 2012-02-7, 2012-02-05, 201...
$ type          <fct> Restricted Holiday, Observance, Restricted Holiday, Restricted Holiday, Observance, Gazetted Holiday, Restricted Holiday,...
$ description.  <fct> New Year's Day;, Guru Govind Singh Jayanti;, Pongal;, Makar Sankranti;, Chinese New Year;, Republic Day;, Vasant Panchami...
Observations: 1,218
Variables: 2
$ date          <fct> jan/13;5, feb/13;5, Mar 2013;5, apr/13;5, May 2013;5, jun/13;5, jul/13;6, aug/13;6, sep/13;6, Oct 2013;6, nov/13;6, dec/1...
$ dcoilwtico..  <fct> 708.32;3.29%, 784.94;1.34%, 575.84;-3.61%, 375.04;-3.60%, 468.66;1.74%, 817.69;6.38%, 292.38;8.16%, 836.67;8.65%, 926.83;...
Observations: 2,091,618
Variables: 5
$ date        <fct> 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 20...
$ store_nbr   <int> 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2...
$ item_nbr    <int> 103665, 214381, 315179, 414752, 557408, 657869, 781797, 859529, 953609, 1062374, 115267, 168930, 229368, 305341, 358561, 4...
$ unit_sales  <dbl> 7.000, 5.000, 4.000, 1.000, 3.000, 1.000, 2.000, 1.000, 1.000, 10.601, 7.000, 25.000, 6.000, 28.000, 1.000, 1.000, 3.000, ...
$ onpromotion <fct> , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , , ,
Observations: 83,488
Variables: 3
$ date          <fct> 2013-01-01, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2...
$ store_nbr     <int> 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 37...
$ transactions  <int> 770, 2111, 2358, 3487, 1922, 1903, 2143, 1874, 3250, 2940, 1293, 3547, 1362, 1102, 2002, 1622, 1167, 1580, 1635, 1369, 13...
```

Checking out the categorical data

Make a list of all categorical data elements in the provided files. (HINT: use the glimpse function and look for factors)



Checking out the factors

```
levels(x)
```

- Shows a distinct list of values from a factor variable
- Needs a variable (which contains values)

E.g: levels(carBrand) could return "Ford", "Audi", "BMW", "Mercedes"

Focusing on stores

$ VS [,x]

What are the possible values in the state, type and cluster column?

```
$ store_nbr    <int> 1,
$ Store.name   <fct> BE
$ city         <fct> GU
$ state        <fct> As
$ type         <fct> D,
$ cluster      <int> 1
```

```r
levels(stores$state)
levels(stores$type)
levels(stores$cluster)
```

```
[1] "Assam"         "Bihar"         "Chhattisgarh" "Jharkhand"
[5] "Maharashtra"   "Odisha"        "Sikkim"       "Tripura"
[9] "West Bengal"
[1] "A" "B" "C" "D" "E"
NULL
```

There's a problem

Why does cluster show NULL ?

```
$ cluster      <int> 1
```

UH OH

There's a problem

...but we can fix it!

Turns a vector into a factor (can be any vector, must be sortable and have as.character method)

```
as.factor(x)
```

Try finding the possible values now.

Combining categorical data

Categorical variable nr. 1

Categorical variable nr. 1

	Cat 1	Cat 2	Cat 3	…	…	…
Cat 1	Freq	Freq	Freq	Freq	Freq	Freq
Cat 2	Freq	Freq	Freq	Freq	Freq	Freq
Cat 3	Freq	Freq	Freq	Freq	Freq	Freq
…	Freq	Freq	Freq	Freq	Freq	Freq
…	Freq	Freq	Freq	Freq	Freq	Freq
…	Freq	Freq	Freq	Freq	Freq	Freq
…	Freq	Freq	Freq	Freq	Freq	Freq

Combining categorical data

Create a frequency table with two categorical variables (also called contingency table)

```
table(...)
```

Requires one or more objects (i.e categorical columns)

Combining categorical data

Use the table function to show how many of each store type are in each city

	Assam	Bihar	Chhattisgarh	Jharkhand	Maharashtra	Odisha	Sikkim	Tripura	West Bengal
A	0	0	0	0	0	0	0	0	9
B	0	0	2	0	2	1	0	0	3
C	0	0	2	5	2	1	0	0	5
D	4	4	0	0	0	3	1	1	5
E	0	0	0	0	0	0	0	0	4

Exploring categorical data

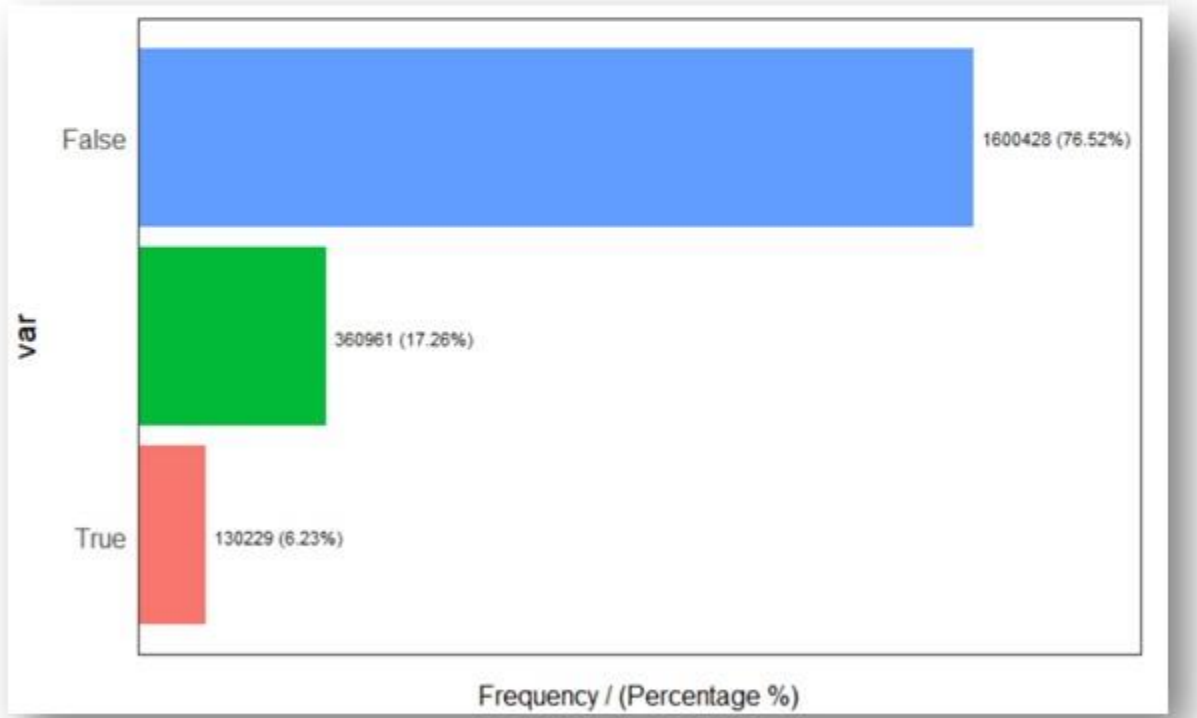
Draws a frequency table based on the object you provide it with and if possible, draws it.

```
freq(x, digits = 1, cum = FALSE, total = FALSE, exclude = NULL,
  sort = "", valid = !(NA %in% exclude), levels = c("prefixed",
  "labels", "values"), na.last = TRUE)
```

Using one variable (so no cross-table)

Exploring categorical data

How many levels does the onPromotion column have in the train set? What are the counts for each?



Exploring numerical data

	Discrete		Continuous
	Nominal	Ordinal	Interval/ratio
Central tendency	Mode	Mode Median	Mode Median Arithmetic mean Geometric mean
Variability		Range Quartiles Interquartile range	Range Quartiles Interquartile range Variance Standard deviation Coefficient of variance
Central tendency and skewness		Boxplot	Boxplot
Correlation			Correlation coefficient Covariance

Exploring numerical data

Bunch of functions each for a specific statistic (not exhaustive)

```
mean(x, …)
range(…, na.rm = FALSE)
median(x, na.rm = FALSE, …)
sd(x, na
Mode(x, na.rm = FALSE)
```

Requires DescTools package

Arithmetic mean only (you'll have to use a custom function for geometric mean)

Exploring numerical data

Profiling_num (requires funModeling package)

```
profiling_num(data)
```

variable <chr>	mean <dbl>	std_dev <dbl>	variation_coef <dbl>	p_01 <dbl>	p_05 <dbl>	p_25 <dbl>	p_50 <dbl>	p_75 <dbl>	p_95 <dbl>
unit_sales	8.53666	20.2157	2.368105	1	1	2	4	9	29

Exploring numerical data

What is the average unit sales across the whole train set?

What is the standard deviation of oil prices?

At which date did we get the highest amount of transactions?

Table of Contents

In this chapter you will learn:

- The basics of ggplot


Time to start drawing

Visualizing categorical data

...using ggplot2!

ggplot2

ggplot(data, aes()) + layer_name()

Every ggplot requires:

- The data you wish to visualize (data)
- The variables you are interested in that are inside of that data (aes)
- The way you want to model those variables (layer_name)

Visualizing categorical data

...using ggplot2!

ggplot2

ggplot(data, aes()) + layer_name()

A ggplot can also have:

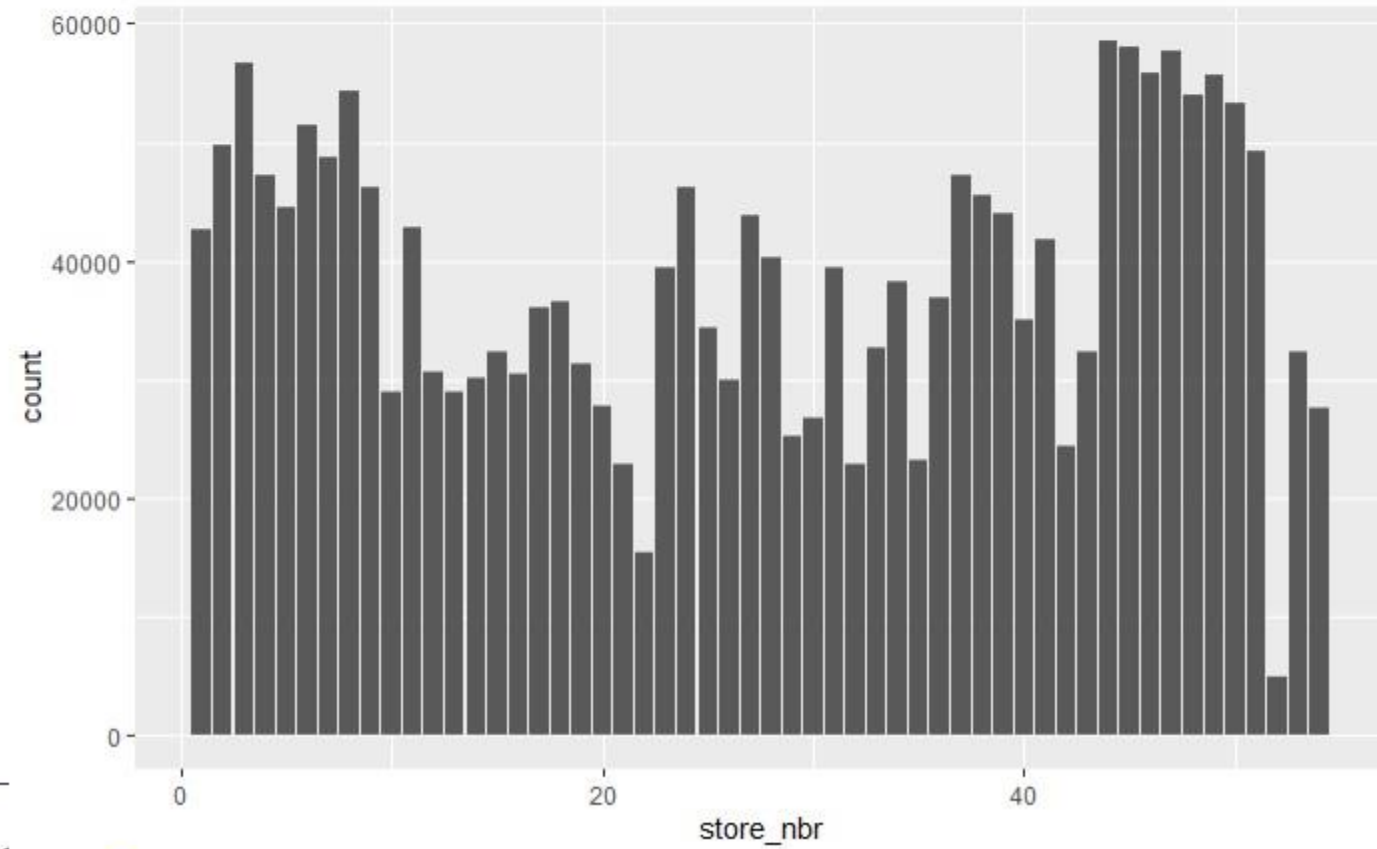
- Statistical measures (stat_)
- Position adjustment (position_)
- Scales (scale_, labs(), xlab(), ylab(), ...)
- Facetting
- Theming

Visualizing categorical data

...using an example!

ggplot2

```
ggplot(train, aes(store_nbr) + geom_bar()
```



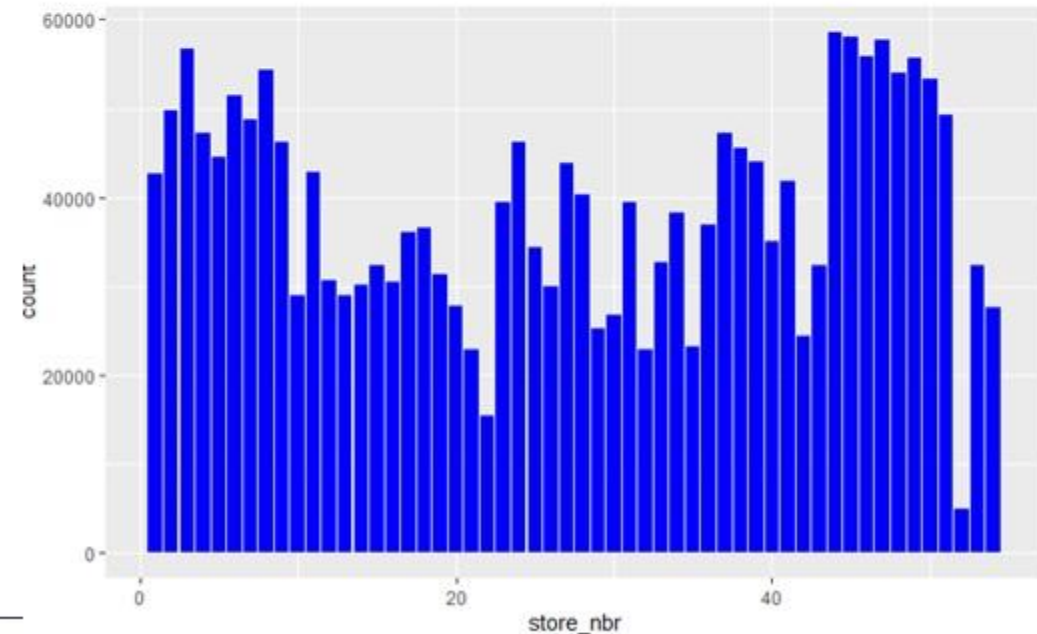
Visualizing categorical data

...using an example!

ggplot2

```
ggplot(train, aes(store_nbr) + geom_bar(fill = "blue")
```

Try turning the bars themselves blue. (HINT: use the fill parameter)



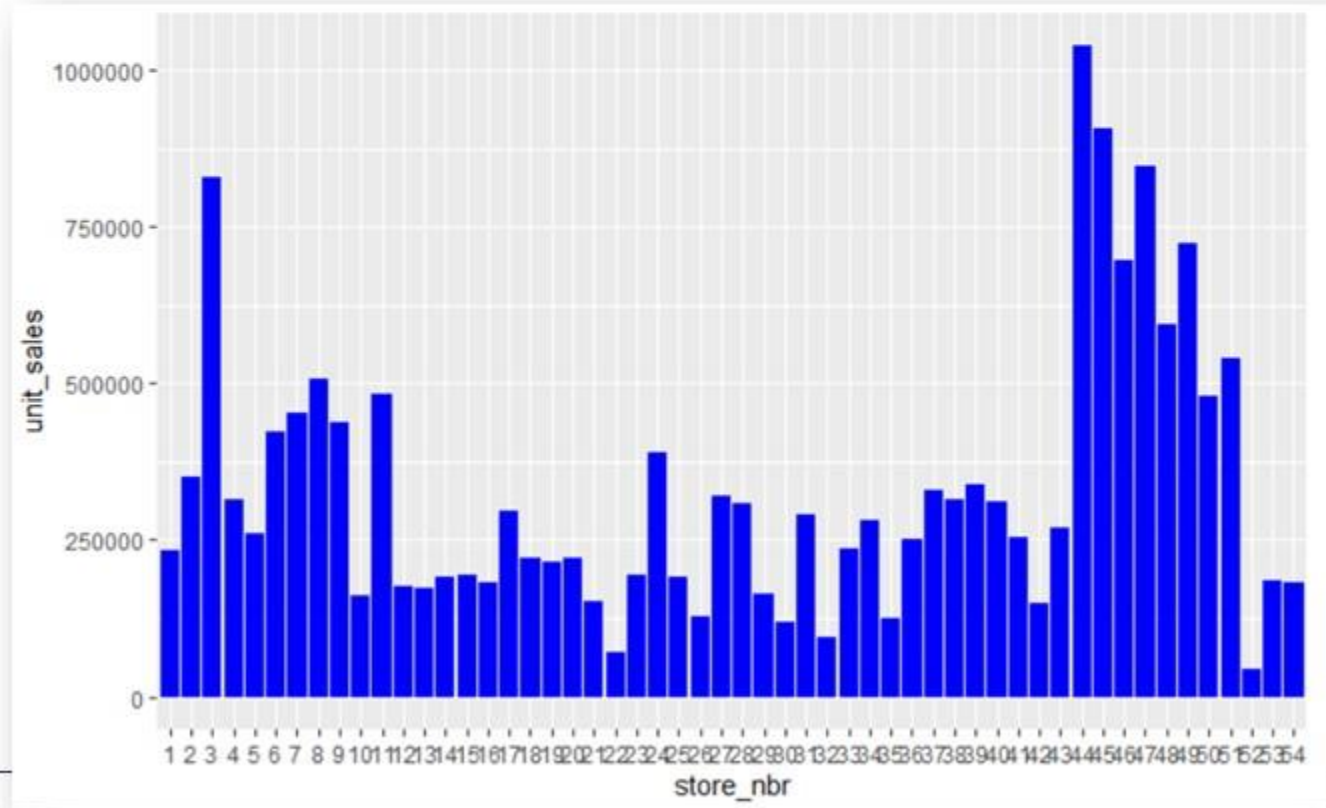
Visualizing categorical data

...using an example!

ggplot2

```
ggplot(train, aes(store_nbr, unit_sales)) + geom_bar(stat = "identity", fill = "blue"))
```

Can you spot the difference?



Visualizing categorical data

...using an example!

ggplot2

```
ggplot(train, aes(store_nbr, unit_sales)) + geom_bar()
```

Now try this.

This won't work, because geom_bar has stat_count() as default!

A lot goes on behind the scenes!

Visualizing categorical data

ggplot2

Ggplot has a **TON** of possible visualizations, each with their own parameters.

geom_abline() geom_hline() geom_vline()	Reference lines: horizontal, vertical, and diagonal
geom_bar() geom_col() stat_count()	Bar charts
geom_bin2d() stat_bin_2d()	Heatmap of 2d bin counts
geom_blank()	Draw nothing
geom_boxplot() stat_boxplot()	A box and whiskers plot (in the style of Tukey)
geom_contour() stat_contour()	2d contours of a 3d surface
geom_count() stat_sum()	Count overlapping points
geom_density() stat_density()	Smoothed density estimates
geom_density_2d() stat_density_2d()	Contours of a 2d density estimate
geom_dotplot()	Dot plot
geom_errorbarh()	Horizontal error bars
geom_hex() stat_bin_hex()	Hexagonal heatmap of 2d bin counts
geom_freqpoly() geom_histogram() stat_bin()	Histograms and frequency polygons
geom_jitter()	Jittered points

geom_crossbar() geom_errorbar() geom_linerange() geom_pointrange()	Vertical intervals: lines, crossbars & errorbars
geom_map()	Polygons from a reference map
geom_path() geom_line() geom_step()	Connect observations
geom_point()	Points
geom_polygon()	Polygons
geom_qq_line() stat_qq_line() geom_qq() stat_qq()	A quantile-quantile plot
geom_quantile() stat_quantile()	Quantile regression
geom_ribbon() geom_area()	Ribbons and area plots
geom_rug()	Rug plots in the margins
geom_segment() geom_curve()	Line segments and curves
geom_smooth() stat_smooth()	Smoothed conditional means
geom_spoke()	Line segments parameterised by location, direction and distance
geom_label() geom_text()	Text
geom_raster() geom_rect() geom_tile()	Rectangles
geom_violin() stat_ydensity()	Violin plot

https://ggplot2.tidyverse.org/reference/

Visualizing categorical data

ggplot2

ggplot(data, aes()) + layer_name()

To get the aes-options of a geom:

```
env <- asNamespace("ggplot2")
req_geom <- ls(envir = env, pattern = "^GeomPoint")
req_geom <- mget(req_geom, env)
req_aes <- map(all_Geoms, ~.$aesthetics())


head(req_aes)
```

Visualizing categorical data

… using an exercise!

ggplot2

Make a stacked bar chart and show the store types per state

ggplot(data, mapping = aes() + layer_name())

What do you need?

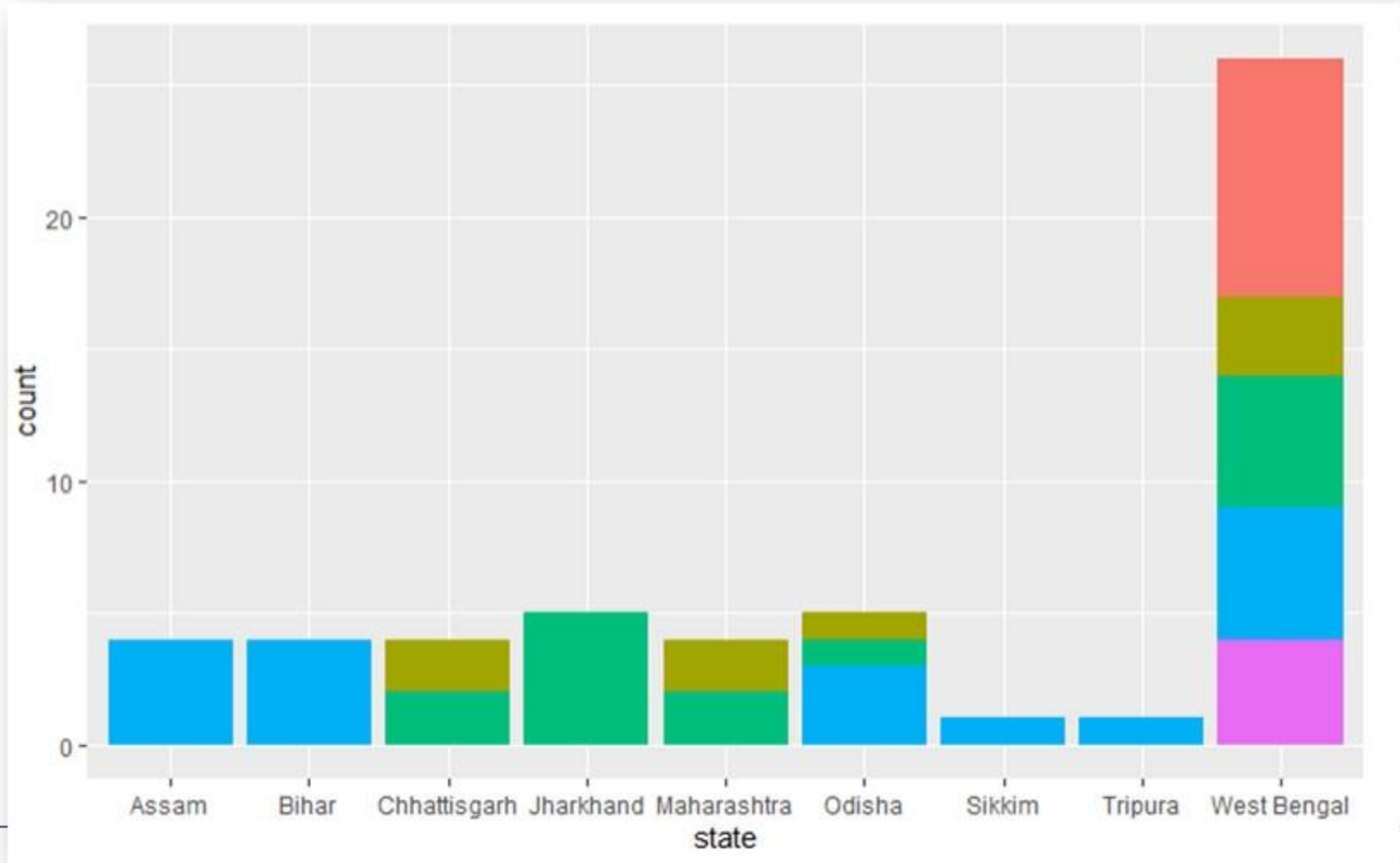
- The data: ‘stores’ table
- The variables: ‘type’ and ‘state’
- The visual: geom_bar()

HINT: for each bar (per state) you are **filling** it up with the count per type of store

Visualizing categorical data

... using an exercise!

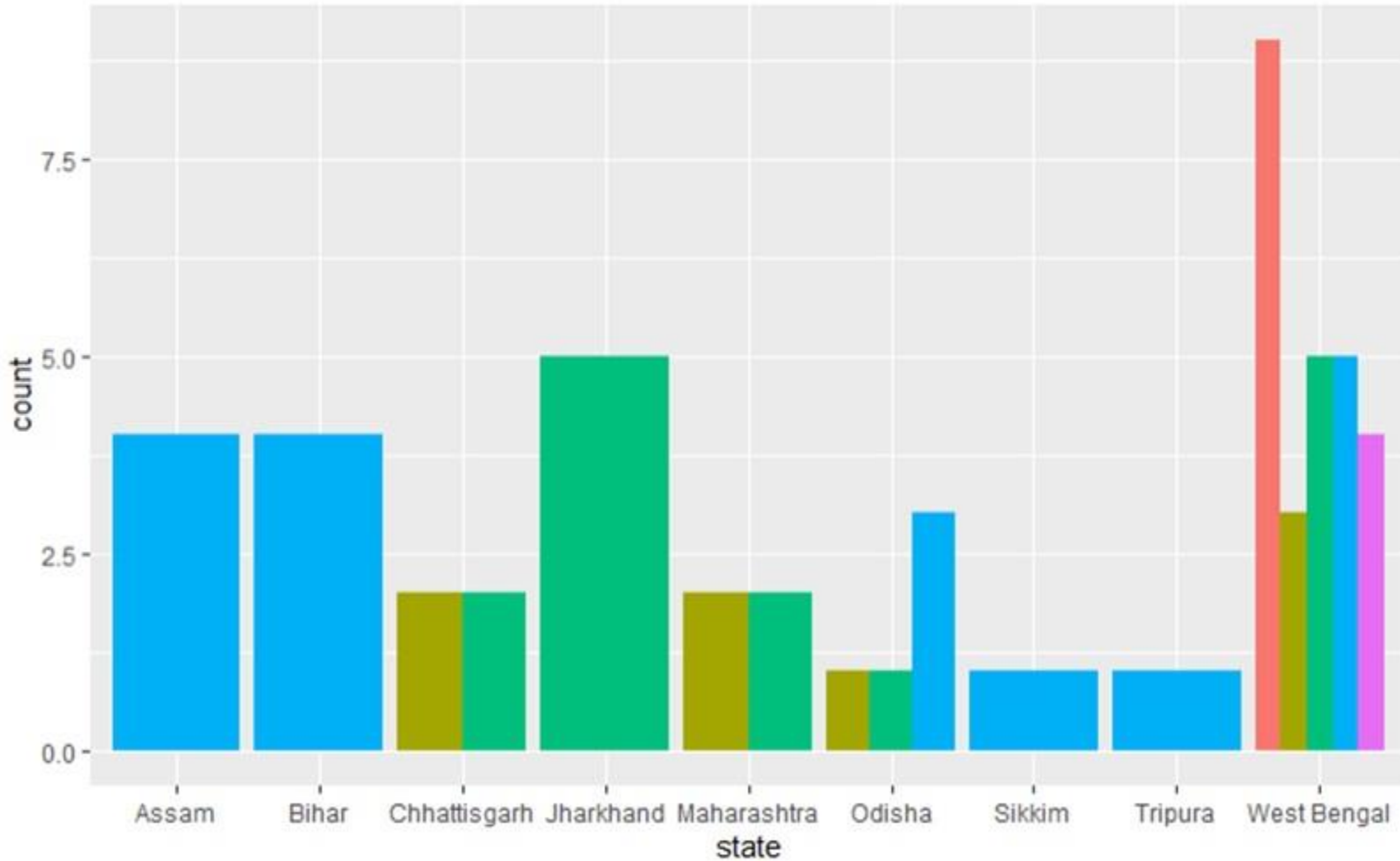
ggplot2



Visualizing categorical data

... using an exercise!

ggplot2



Visualizing categorical data

... using an exercise!

ggplot2

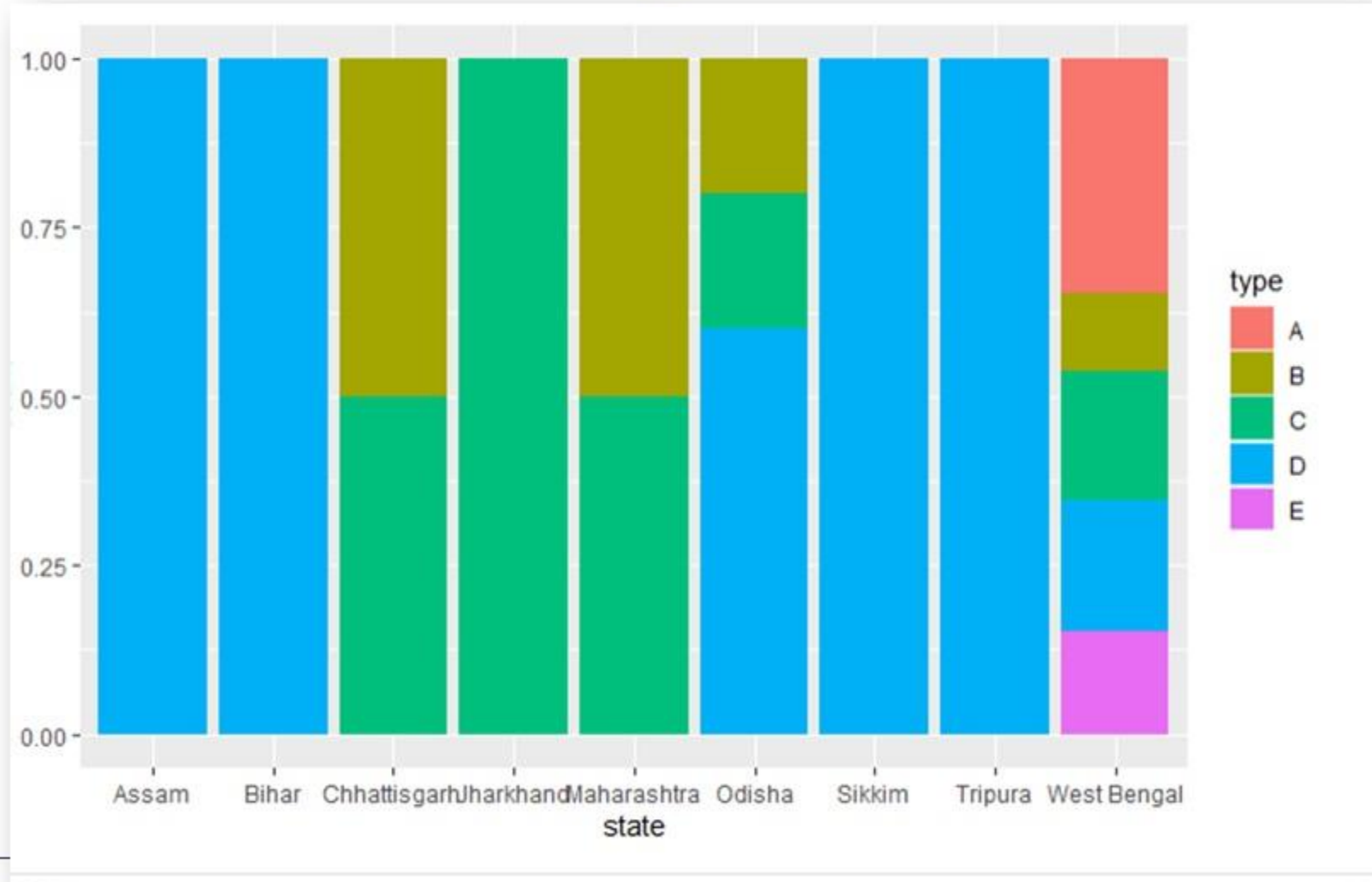


Table of Contents

In this chapter you will learn:

- The basics of ggplot

Remember this?

Interval

°C

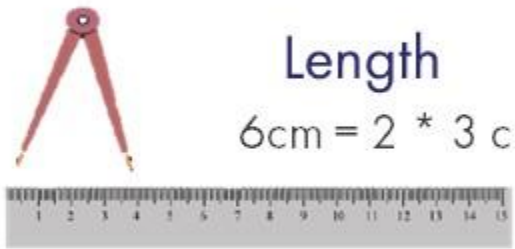
20° is not 4 * 5°

Clock time

4 p.m. is not twice as late as 2 p.m.

Natural origin: No
Meaningful ratios: No

Ratio



Length

6cm = 2 * 3 cm

°Kelvin

Natural origin: Yes
Meaningful ratios: Yes

Checking out the data (again)

Let's first take a look at the numerical data we have available. (HINT: use the glimpse function and look for int, dbl and num)

```
Observations: 54
Variables: 6
$ store_nbr   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, ...
$ Store.name <fct> BB - FC-GUWAHATI-BHANAGAGHAR-CITY SQUARE  G.S ROAD, BB-JOHRAT-MOUZA- NAGARMHAL-PMS / B&A COMPLEX , BB-SILCHAR-PARGANA-BARAK...
$ city        <fct> GUWAHATI, JOHRAT, SILCHAR, GUWAHATI, PATNA, PATNA, BHAGALPUR, PATNA, RAIPUR, BILASPUR, BHILAI, RAIPUR, DHANBAD, RANCHI, JAM...
$ state       <fct> Assam, Assam, Assam, Assam, Bihar, Bihar, Bihar, Bihar, Chhattisgarh, Chhattisgarh, Chhattisgarh, Chhattisgarh, Jharkhand, ...
$ type        <fct> D, D, D, D, D, D, D, D, B, C, B, C, C, C, C, C, C, B, C, B, B, C, D, D, D, D, D, E, E, C, B, C, C, B, C, E, D, D, B, C, D, ...
$ cluster     <int> 13, 13, 8, 9, 4, 13, 8, 8, 6, 15, 6, 15, 15, 7, 15, 3, 12, 16, 15, 6, 6, 7, 9, 1, 1, 10, 1, 10, 10, 3, 10, 3, 3, 6, 3, 10, ...
Observations: 401
Variables: 3
$ date         <fct> 2012-01-01, 2012-01-05, 2012-01-14, 2012-01-15, 2012-01-05, 2012-01-26, 2012-01-28, 2012-02-5, 2012-02-7, 2012-02-05, 201...
$ type         <fct> Restricted Holiday, Observance, Restricted Holiday, Restricted Holiday, Observance, Gazetted Holiday, Restricted Holiday,...
$ description. <fct> New Year's Day;, Guru Govind Singh Jayanti;, Pongal;, Makar Sankranti;, Chinese New Year;, Republic Day;, Vasant Panchami...
Observations: 60
Variables: 2
$ date  <fct> 2013-01-01, 2013-02-01, 2013-03-01, 2013-04-01, 2013-05-01, 2013-06-01, 2013-07-01, 2013-08-01, 2013-09-01, 2013-10-01, 2013-11-...
$ price <dbl> 5708.32, 5784.94, 5575.84, 5375.04, 5468.66, 5817.69, 5292.38, 5836.67, 5926.83, 5497.77, 5435.78, 5534.28, 5344.00, 5529.28, 53...
Observations: 2,091,618
Variables: 5
$ id         <int> 0, 60, 120, 180, 240, 300, 360, 420, 480, 540, 600, 660, 720, 780, 840, 900, 960, 1020, 1080, 1140, 1200, 1260, 1320, 1380,...
$ date       <fct> 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 2013-01-01, 201...
$ store_nbr  <int> 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,...
$ item_nbr   <int> 103665, 214381, 315179, 414752, 557408, 657869, 781797, 859529, 953609, 1062374, 115267, 168930, 229368, 305341, 358561, 41...
$ unit_sales <dbl> 7.000, 5.000, 4.000, 1.000, 3.000, 1.000, 2.000, 1.000, 1.000, 10.601, 7.000, 25.000, 6.000, 28.000, 1.000, 1.000, 3.000, 1...
Observations: 83,488
Variables: 3
$ date         <fct> 2013-01-01, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2013-01-02, 2...
$ store_nbr    <int> 25, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 37...
$ transactions <int> 770, 2111, 2358, 3487, 1922, 1903, 2143, 1874, 3250, 2940, 1293, 3547, 1362, 1102, 2002, 1622, 1167, 1580, 1635, 1369, 13...
```

Checking out the numbers

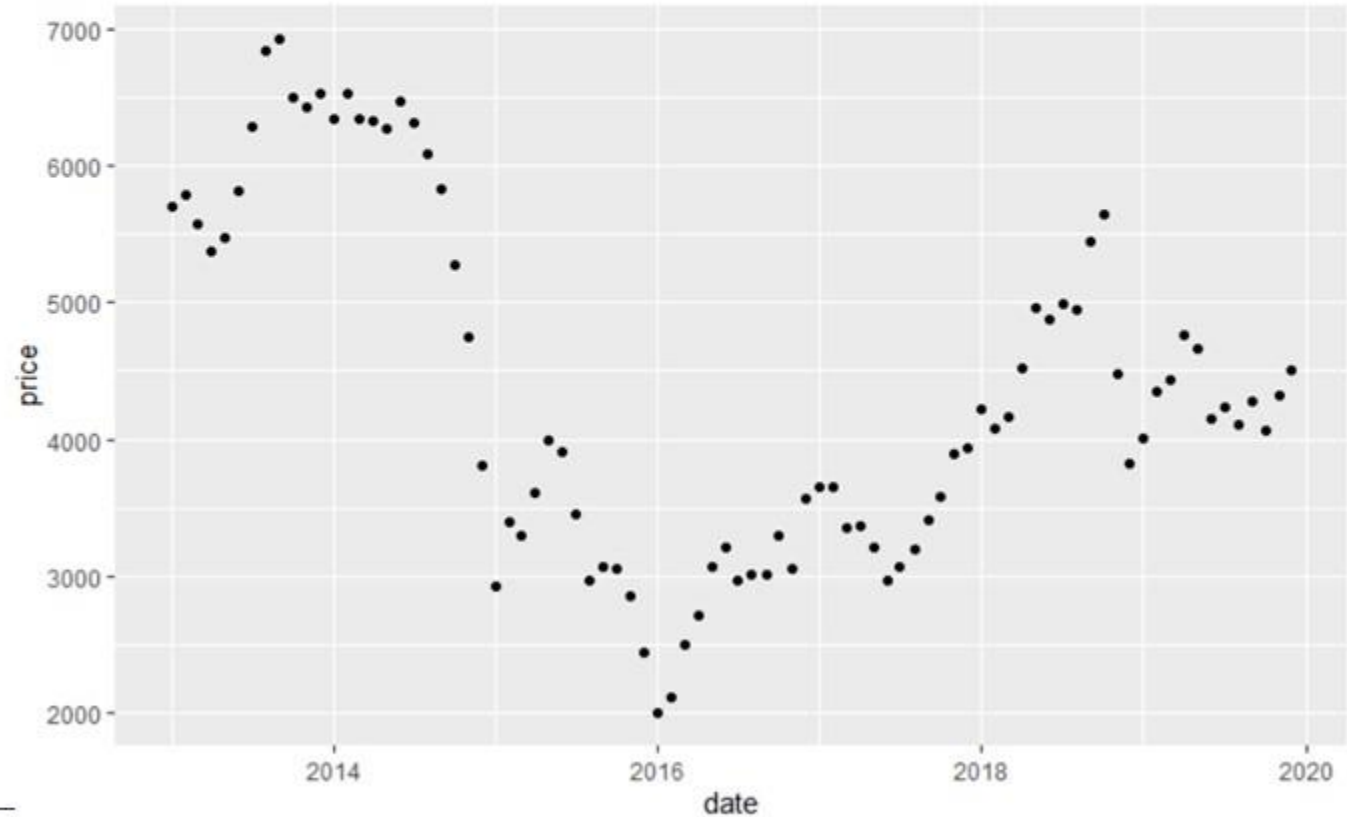
```
geom_point(mapping = NULL, data = NULL, stat = "identity",
  position = "identity", ..., na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE)
```

Let's put this to the test!

Step one:
Making the scatter plot

Start by making a scatter plot by adding the x and y axis in the aes parameter.

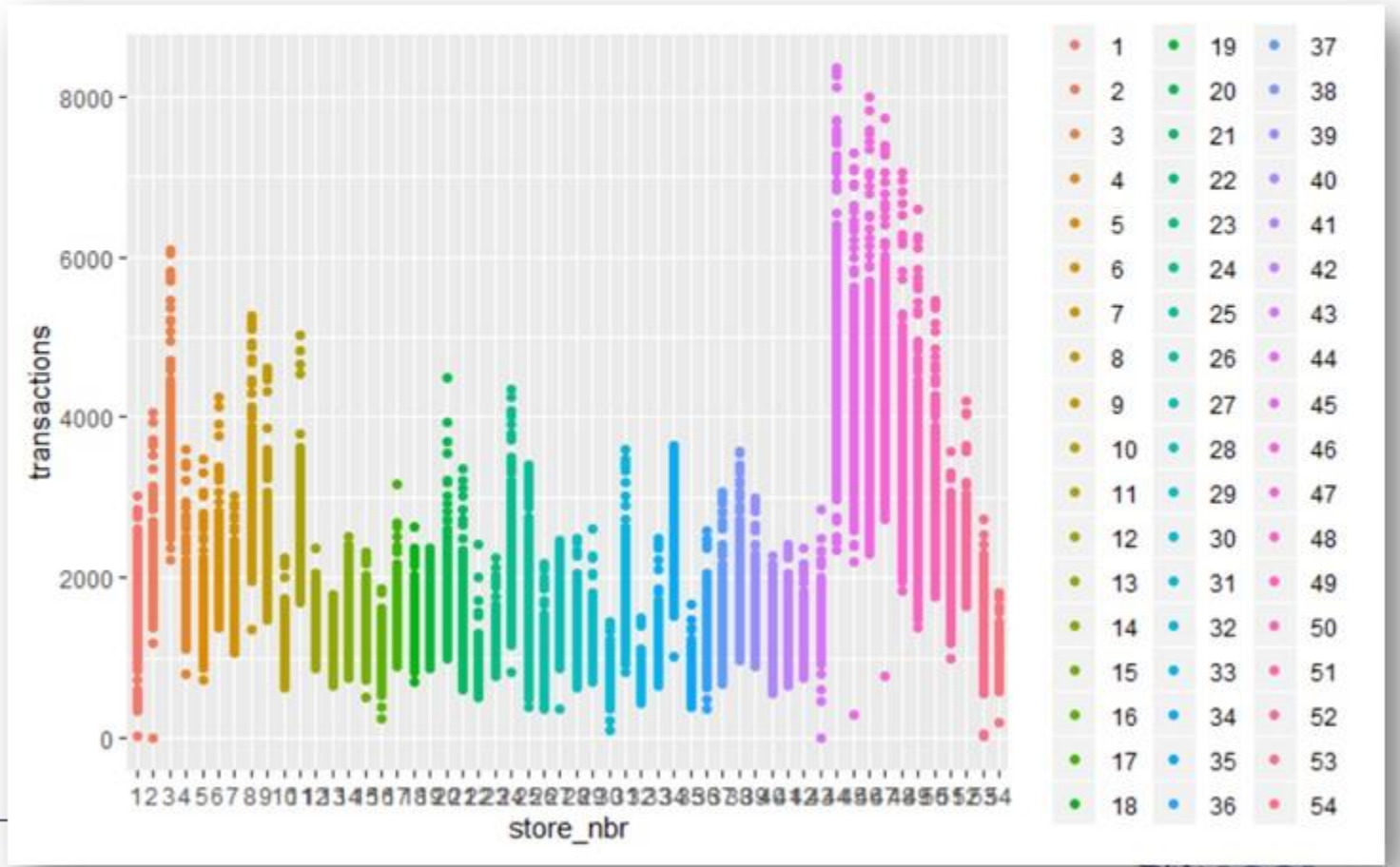
```
```{r pressure, echo=FALSE}
ggplot(oil) + geom_point(aes(date,price))
```
```



Step two:
About scales

Try this:

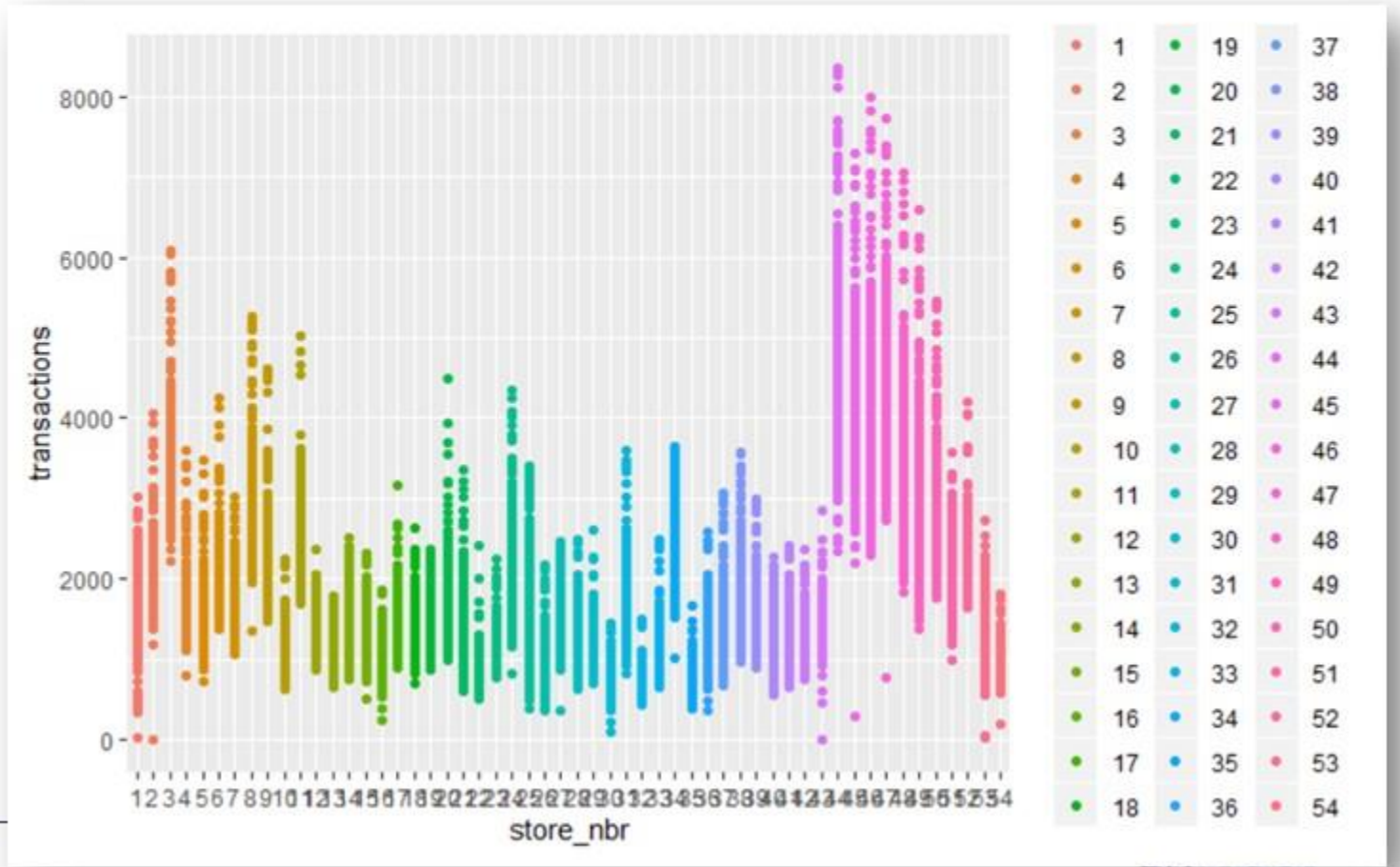
```
ggplot(trans, aes(store_nbr, transactions)) +
geom_point(aes(colour = store_nbr))
```



Step two:
About scales

Now try this:

```
ggplot(trans, aes(store_nbr, transactions)) +
  geom_point(aes(colour = store_nbr)) +
  scale_x_discrete() +
  scale_y_continuous()
```



Step two:
About scales

Behind the screens, R helps you with scales your axes.

Scales

- take your data and turn it into something that you can see, like size, colour, position or shape
- provide the tools that let you read the plot: the axes and legends
- is **required** for every aesthetic used on the plot.

We can play with scales!

Step two:
About scales

Scales

- take your data and turn it into something that you can see, like size, colour, position or shape
- provide the tools that let you read the plot: the axes and legends
- is **required** for every aesthetic used on the plot.

Continuous scales (used to map integer, numeric, and date/time data to x and y position)

scale_*_continuous(), scale_*_log10(), scale_*_sqrt(), scale_*_date() and scale_*_datetime()

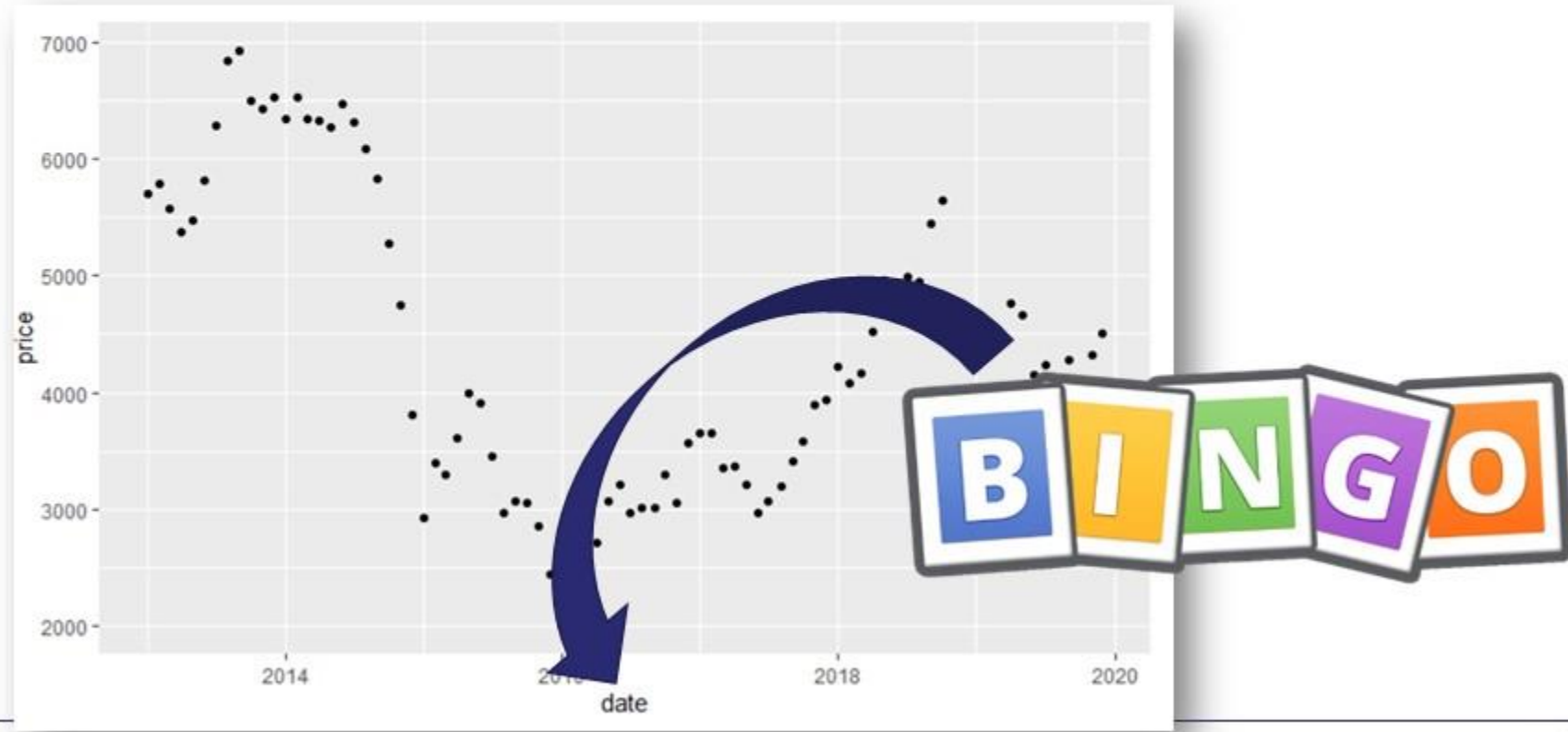
Discrete scales (used to map discrete variables to x and y position)

scale_*_discrete()

There are also colour scales and manual scales (not part of this course).

Step two:
About scales

What kind of variable is our x-axis?



Step two: Adding detail

...using scales!

We can't see or read the yearly tick marks. Add them by using scale_x_date() and the date_breaks parameter.

```
scale_x_date(..., expand = waiver(), breaks = pretty_breaks(),
  minor_breaks = waiver())

scale_y_date(..., expand = waiver(), breaks = pretty_breaks(),
  minor_breaks = waiver())
```

Step two:
Adding detail

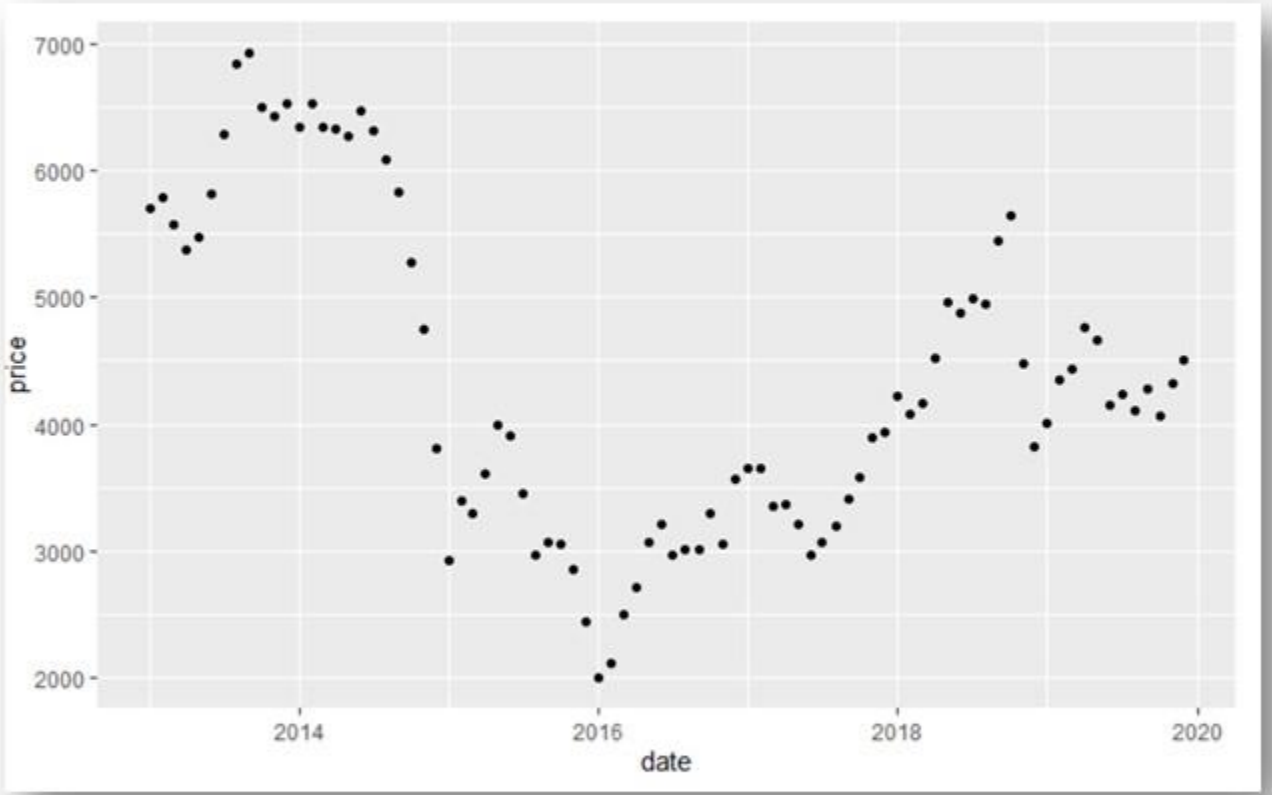
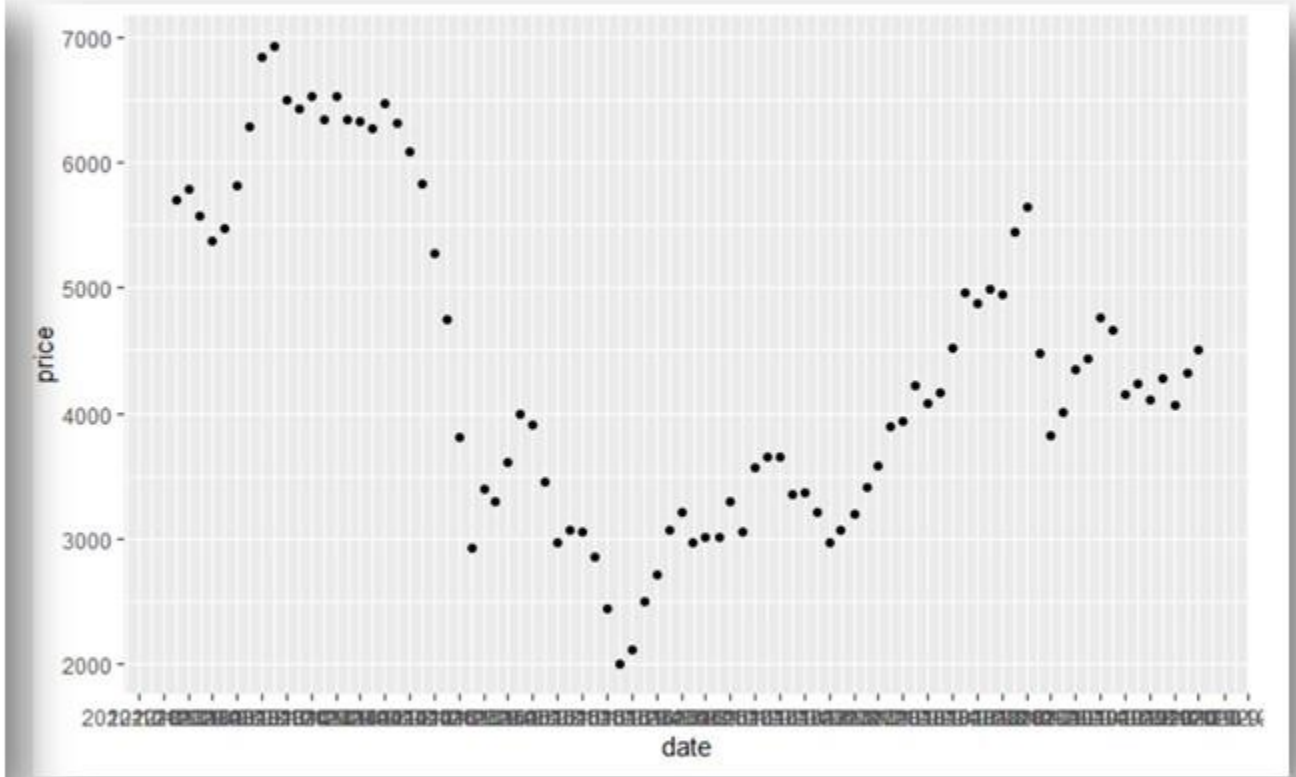
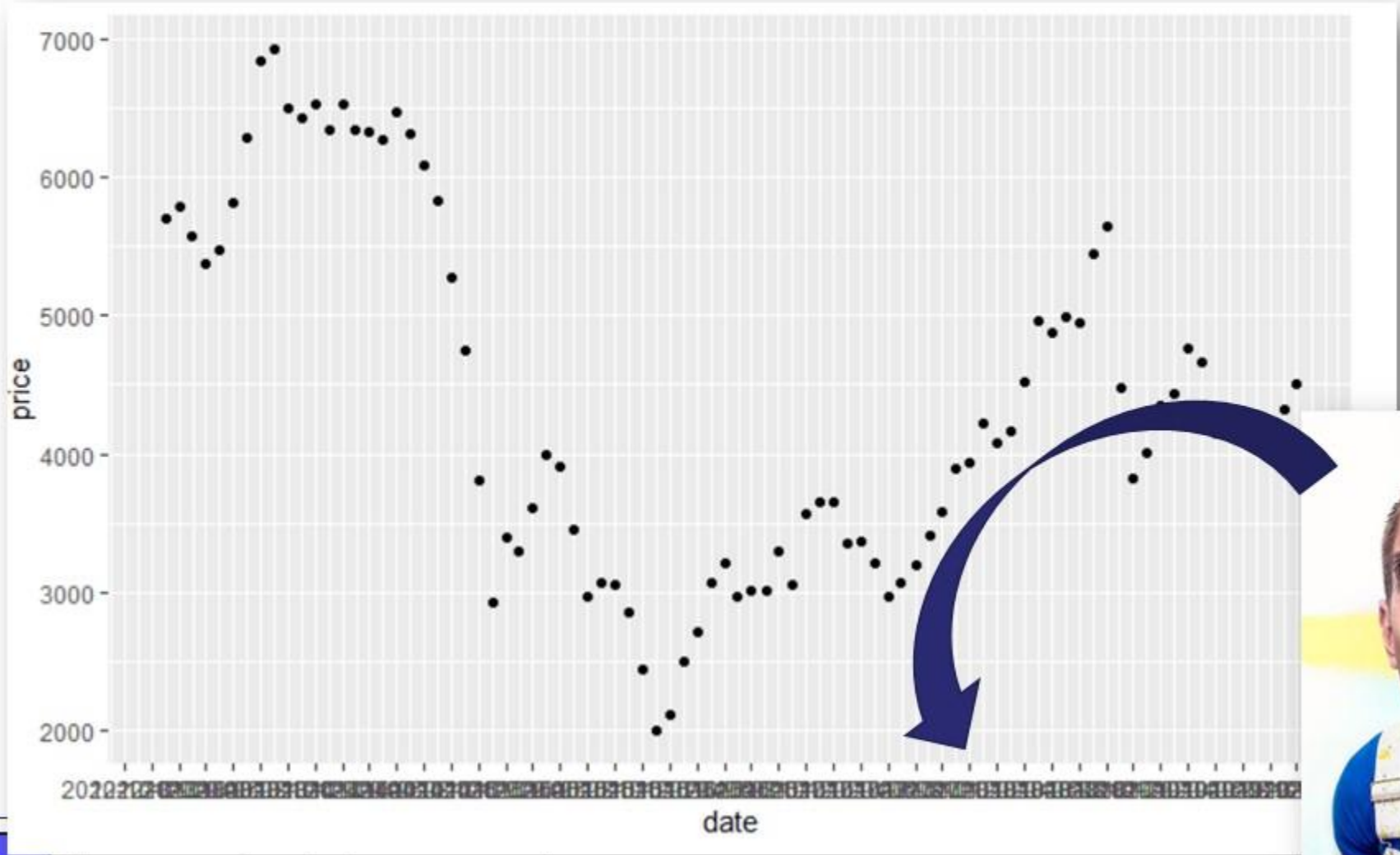
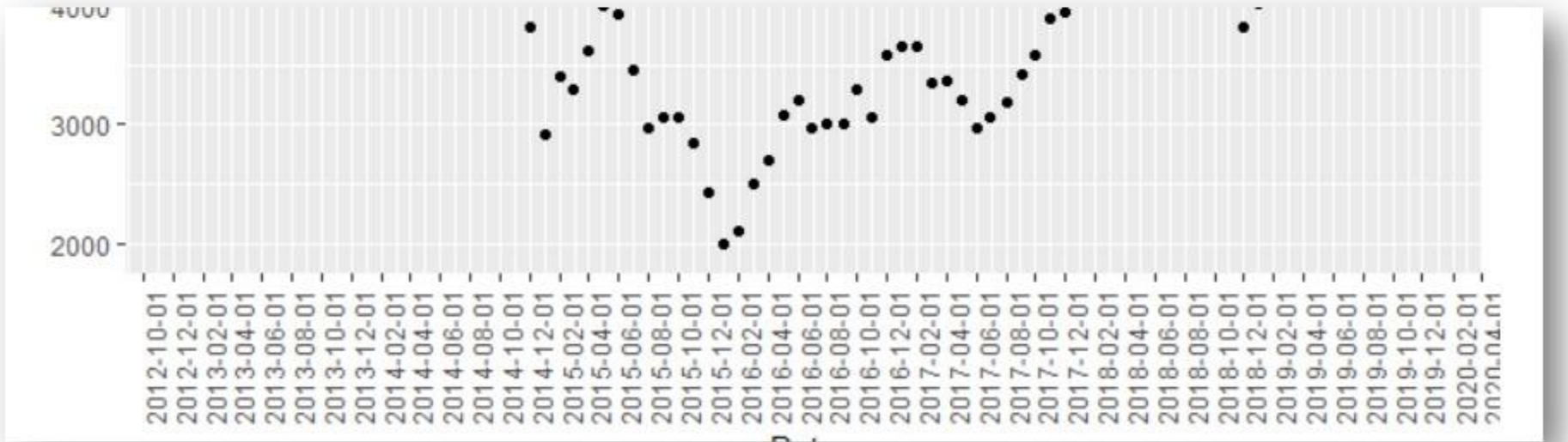
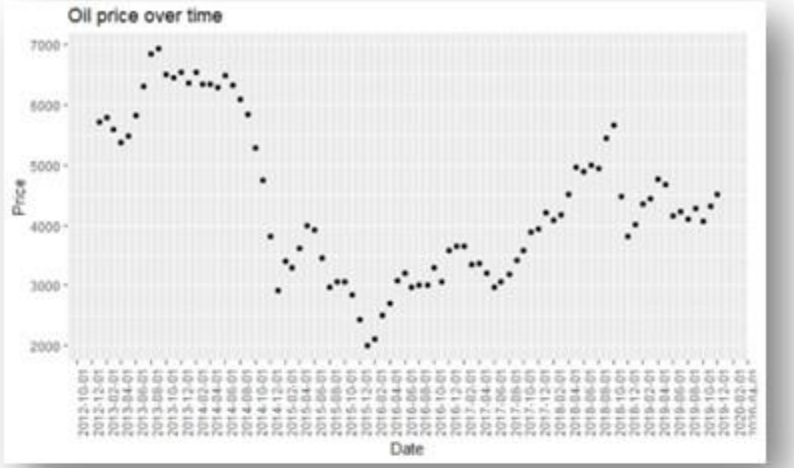
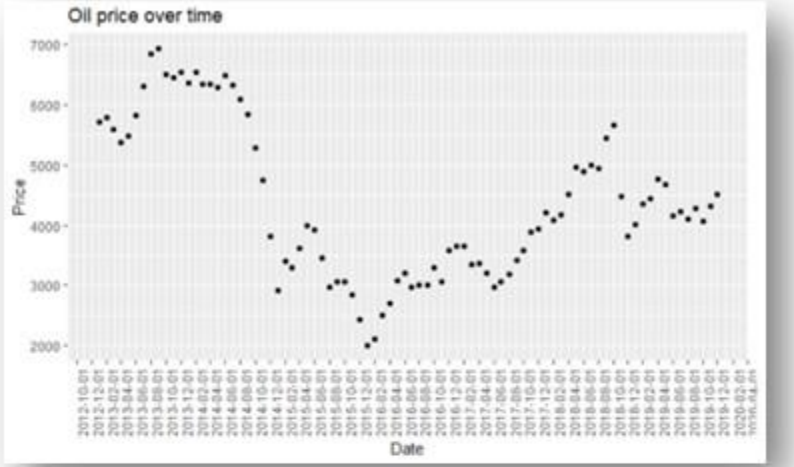
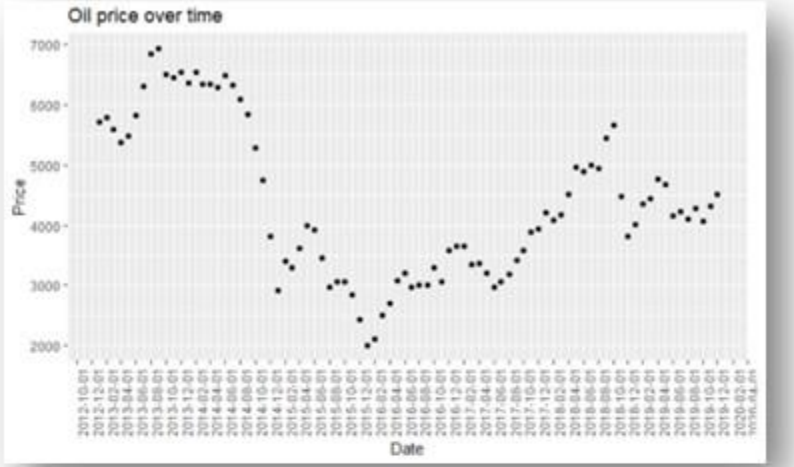
```r
```{r pressure, echo=FALSE}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(date_breaks = "2 months")
```
```

```r
scale_x_date(date_breaks = "2 months")
```

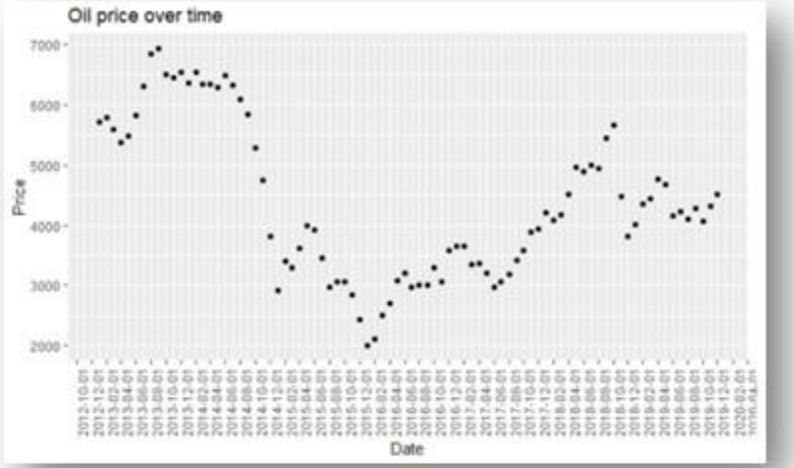
Scale_x_date (or scale_y_date) parameters help transform the axes:

- Breaks = define the width between breaks (date_breaks is more specific)
- Limits = define the start and end point on the respective axis
- Minor_breaks = define the existence (and properties) of minor breaks

Step two: Adding detail

```r
```{r pressure, echo=FALSE}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month"))
```
```





Step two:
Finetuning the details



YUK

Step two:
Finetuning the details

The x-axis is unreadable. Let's fix it using the theme function.

Hang on to your hats!

```
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.ticks.length.x, axis.ticks.length.x.top, axis.ticks.length.x.bottom,
  axis.ticks.length.y, axis.ticks.length.y.left, axis.ticks.length.y.right,
  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,
  legend.spacing, 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, ..., complete = FALSE,
  validate = TRUE)
```

Step two:
Finetuning the details

Depending on the element, it is defined by another function:

- Line = element_line()
- Rect = element_rect()
- Text = element_text()
- Title = element_text()


```
element_text(family = NULL, face = NULL, colour = NULL,
  size = NULL, hjust = NULL, vjust = NULL, angle = NULL,
  lineheight = NULL, color = NULL, margin = NULL, debug = NULL,
  inherit.blank = FALSE)
```

Step two:
Finetuning the details

Use the axis.x.text and element_text functions to angle your x axis text by 90 degrees.

Step two:
Finetuning the details

Use the axis.x.text and element_text functions to angle your x-axis text by 90 degrees.

```
```{r pressure, echo=FALSE}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month")) + theme(axis.text.x =
element_text(angle = 90))
```
```



Step three:
Even finer tuning the details



Labs() is used to modify Axis, Legend and Plot Labels.

```
labs( title = , subtitle = , x = , y = )
```

Adds a title

Adds a subtitle

Adds an x-axis label

Adds a y-axis label

Step three:
Even finer tuning the details



Labs acts as a wrapper for several other functions.

Change title of a graph → ggtitle()

Change x-axis label → xlab()

Change y-axis label → ylab()

Step three:
Even finer tuning the details



- Add a title to your chart
- Change the name of your x-axis to Date (instead of date)
- Change the name of your y-axis to Price (instead of price)

```
```{r pressure, echo=FALSE}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month")) + theme(axis.text.x =
element_text(angle = 90)) + labs(title = "Oil price over time", x = "Date", y = "Price")
```
```

Step three:
Even finer tuning the details



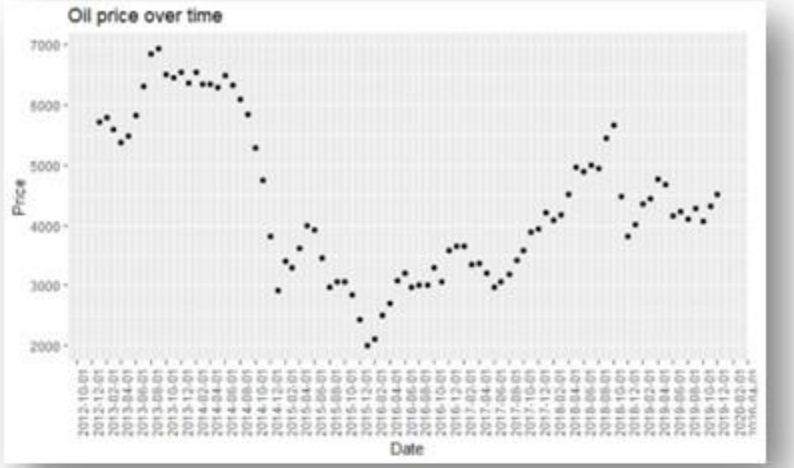
Which means

```{r}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month")) + theme(axis.text.x = element_text(angle = 90)) + ggtitle("Oil price over time") + xlab("Date") + ylab("Price")
```

is identical to

```{r pressure, echo=FALSE}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month")) + theme(axis.text.x = element_text(angle = 90)) + labs(title = "Oil price over time", x = "Date", y = "Price")
```

Step four:
Final step



Let's finalize our plot by reformatting the date (x-axis) so that

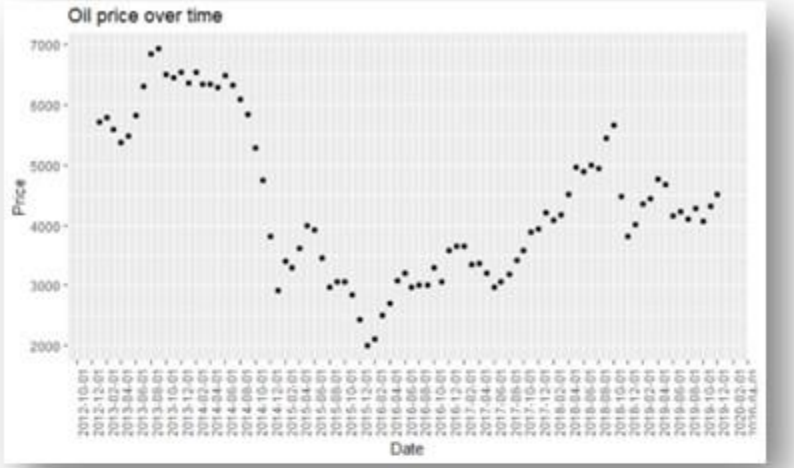
2013-02-01 becomes 2013-02

%Y:	4-digit year
%y:	2-digit year
%m:	2-digit month
%d:	2-digit day of the month
%A:	weekday name
%a:	shorter weekday name
%B:	month name
%b:	shorter month name

Given 2013-02-01:

%d-%m-%Y	would be	01-02-2013
%A %B %Y	would be	Friday February 2013
Etc…		

Step four:
Final step

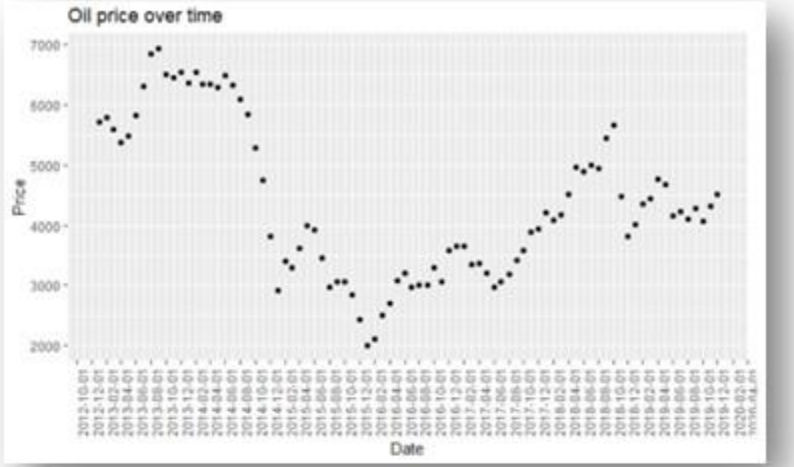


Let's think about this.

```{r}
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month"),
labels=date_format("%Y-%m")) + theme(axis.text.x = element_text(angle = 90)) + ggtitle("Oil price over time") +
xlab("Date") + ylab("Price")
```

Where would we expect to change how the x-axis itself looks like?

Step four:
Final step

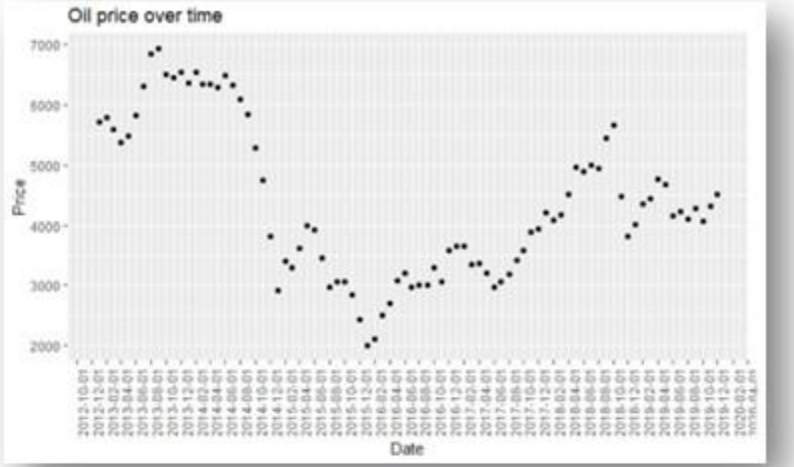


Next question. Which function would you reasonably expect?

Something... something format...

```
date_format(format = "%Y-%m-%d", tz = "UTC")
```

Step four:
Final step



Let's finalize our plot by reformatting the date (x-axis) so that

2013-02-01 becomes 2013-02

```r
ggplot(oil) + geom_point(aes(date,price)) + scale_x_date(breaks = date_breaks(width = "2 month"),
labels=date_format("%Y-%m")) + theme(axis.text.x = element_text(angle = 90)) + labs(title = "Oil price over time", x =
"Date", y = "Price")
```

Table of Contents

In this chapter you will learn:

- How to draw a scatterplot
- How to draw a boxplot

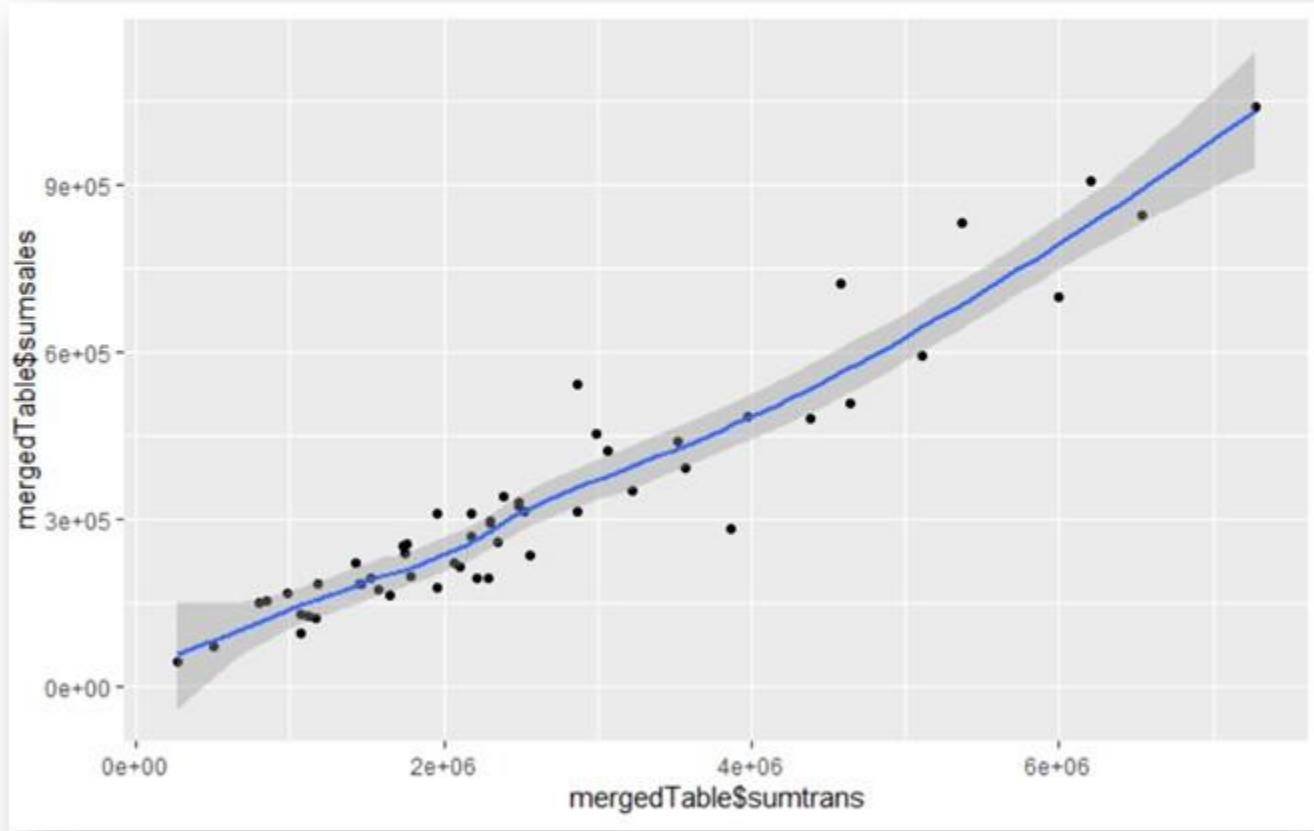
Relations between numerical variables

What better way than the scatter plot?

Make a scatter plot using the mergedTable.

Use geom_point()

Add a line with geom_smooth()



Relations between numerical and categorical variables

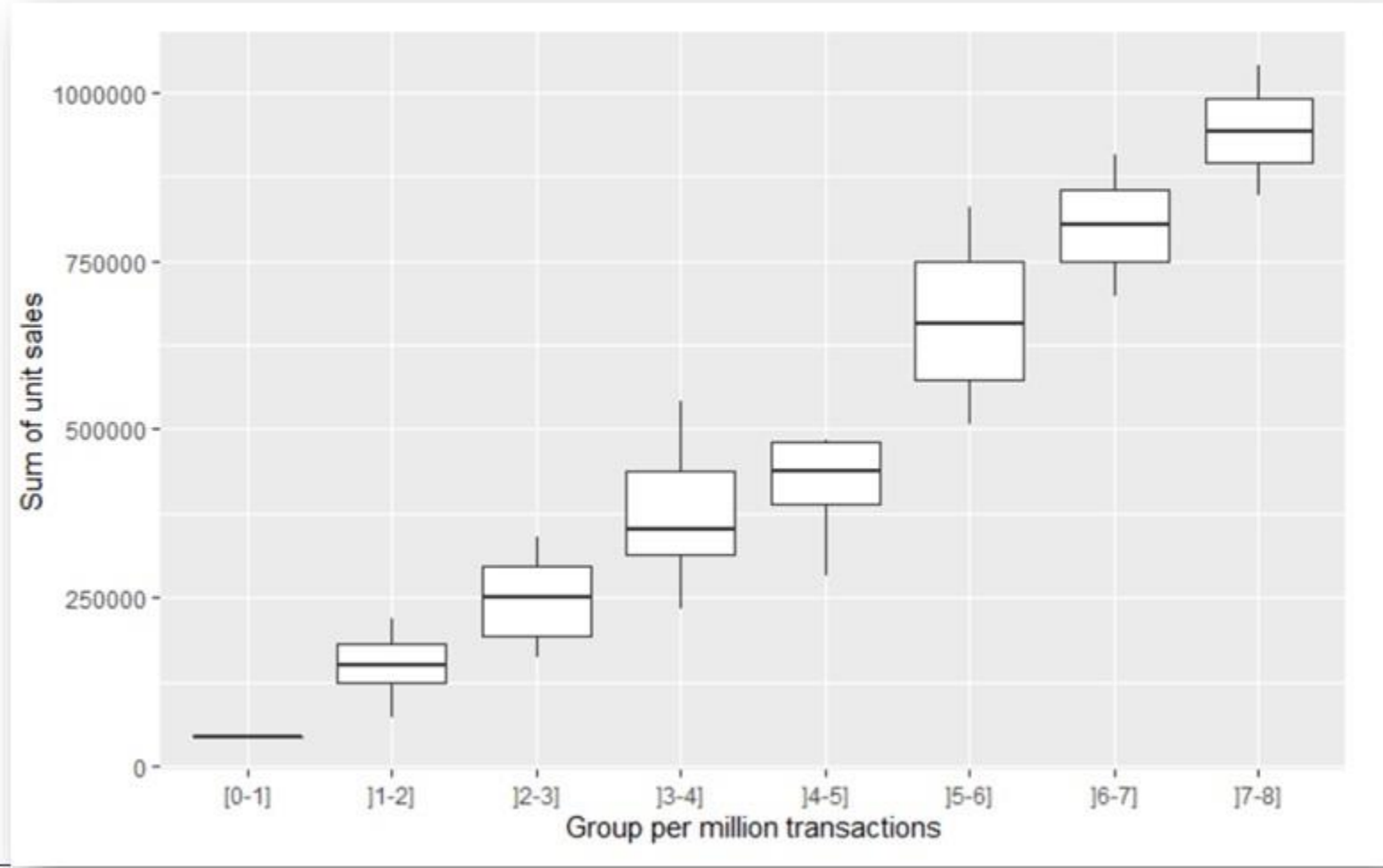
Use the mergedTable to draw a boxplot. These are the requirements:

- Group per million transactions on the x-axis
- Sum of sales on the y-axis
- Change the x label to "Group per million transactions"
- Change the y label to "Sum of unit sales"
- Change the x-axis labels to appropriate ranges (e.g. [0-1] [1-2] etc)

You will need:

- Ggplot
- geom_boxplot
- Cut_width
- Scale_x_discrete
- Xlab
- Ylab

Relations between numerical and categorical variables



Thank you

See you in the next part!