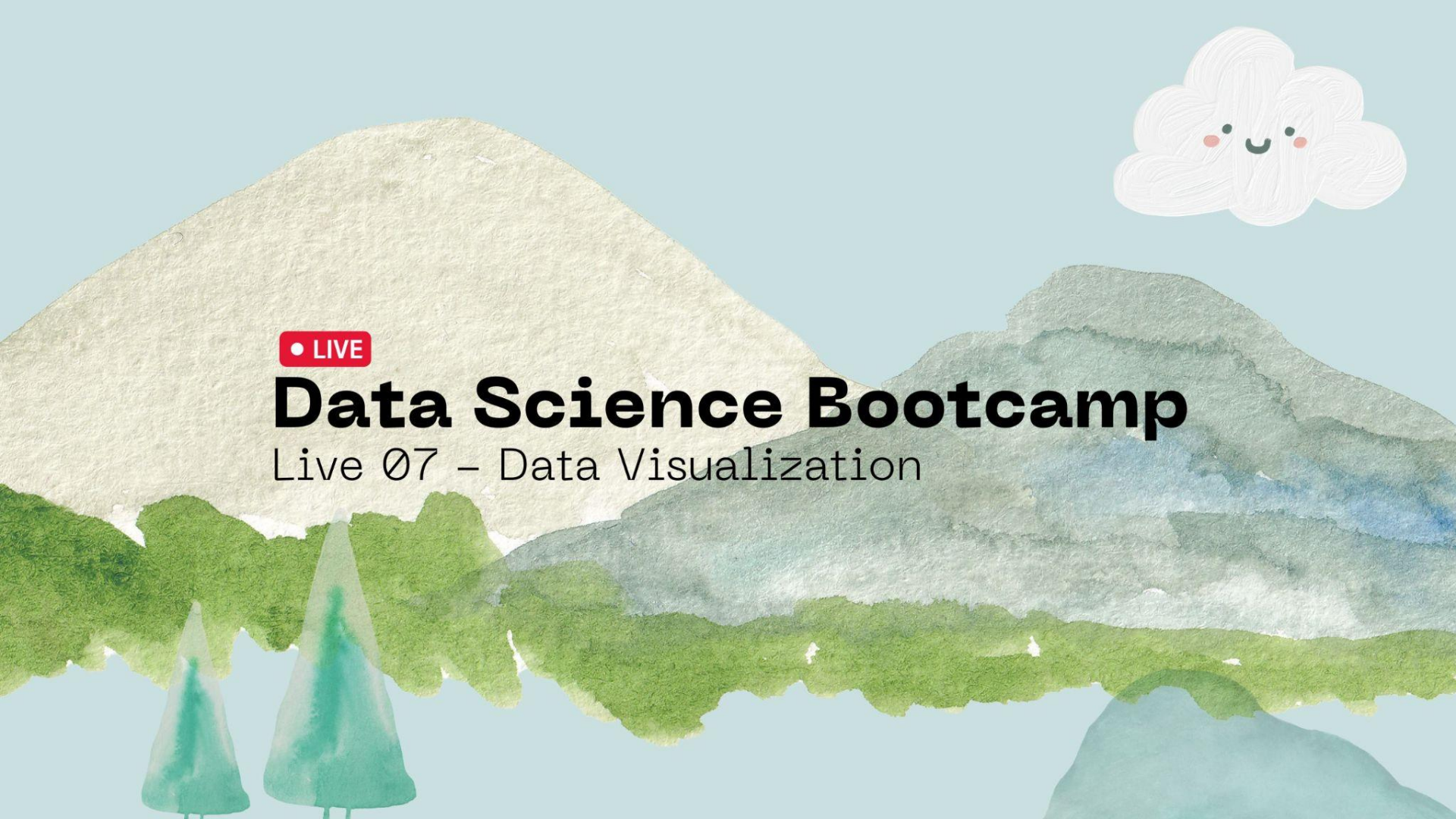

LIVE
Data Science Bootcamp
Live 07 - Data Visualization

Packages we'll cover in this lecture

```
library(tidyverse)
library(glue)
library(plotly)
library(patchwork)
library(corrplot)
library(lubridate)
```

Main package for R developers

install these packages if you haven't :)

Few slides, More practices

3 Data visualisation

3.1 Introduction

> "The simple graph has brought more information to the data analyst's mind than any other device." — John Tukey

This chapter will teach you how to visualise your data using ggplot2. R has several systems for making graphs, but ggplot2 is one of the most elegant and most versatile. ggplot2 implements the **grammar of graphics**, a coherent system for describing and building graphs. With ggplot2, you can do more faster by learning one system and applying it in many places.

If you'd like to learn more about the theoretical underpinnings of ggplot2 before you start, I'd recommend reading "The Layered Grammar of Graphics", http://vita.had.co.nz/papers/layered-grammar.pdf.

3.1.1 Prerequisites

This chapter focusses on ggplot2, one of the core members of the tidyverse. To access the datasets, help pages, and functions that we will use in this chapter, load the tidyverse by running this code:

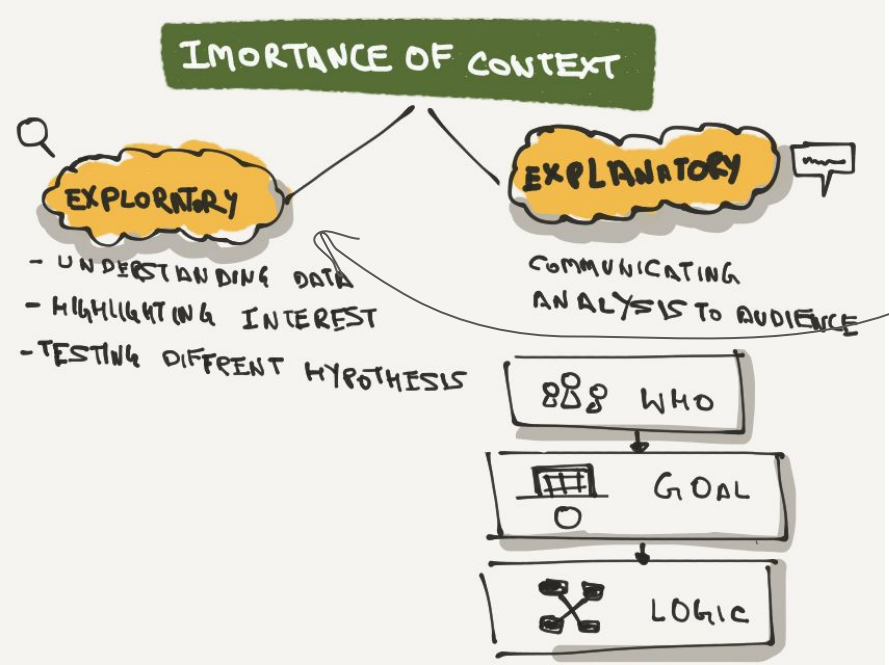
```
library(tidyverse)                                                    Copy
#> ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0
#> ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
#> ✓ tibble  3.0.3     ✓ dplyr   1.0.2
#> ✓ tidyr   1.1.2     ✓ stringr 1.4.0
#> ✓ readr   1.4.0     ✓ forcats 0.5.0
#> ── Conflicts ────────────────────────────────────────── tidyverse_conflicts()
#> ✕ dplyr::filter() masks stats::filter()
#> ✕ dplyr::lag()    masks stats::lag()
```

Welcome | R for Data Science (had.co.nz)

Today we'll learn from Hadley Wickham's book (data science with R)

The author of ggplot2 package

Exploratory vs. Explanatory

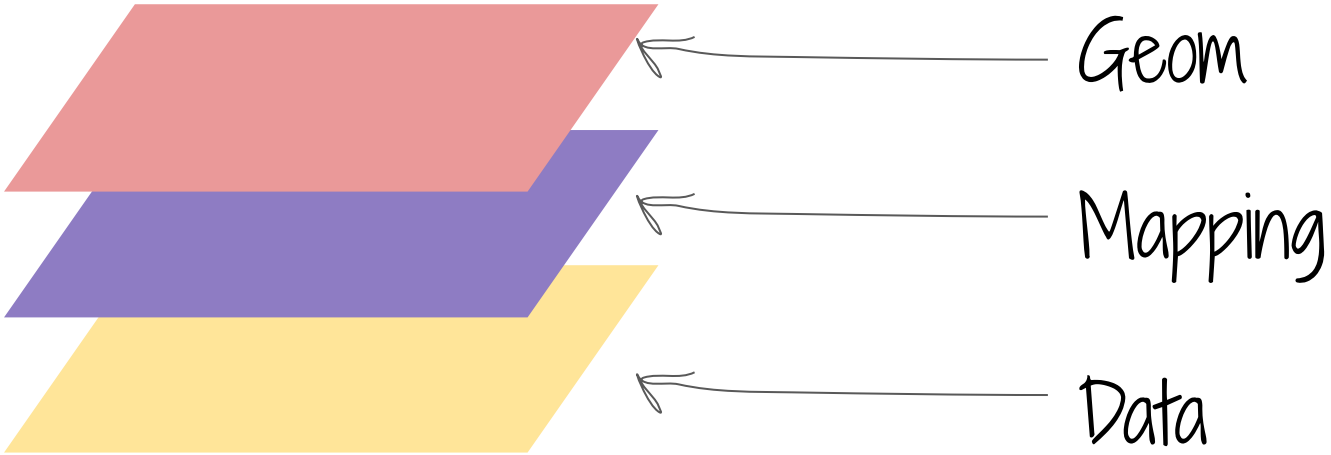


Our lecture focuses on
exploratory

We'll create a lot of plots (very quickly) to find insights and useful patterns

TELL STORIES WITH DATA... We have a lot of data with cumbersome... | by Nihal Walia | UXfools | Medium

ggplot foundation



ggplot foundation

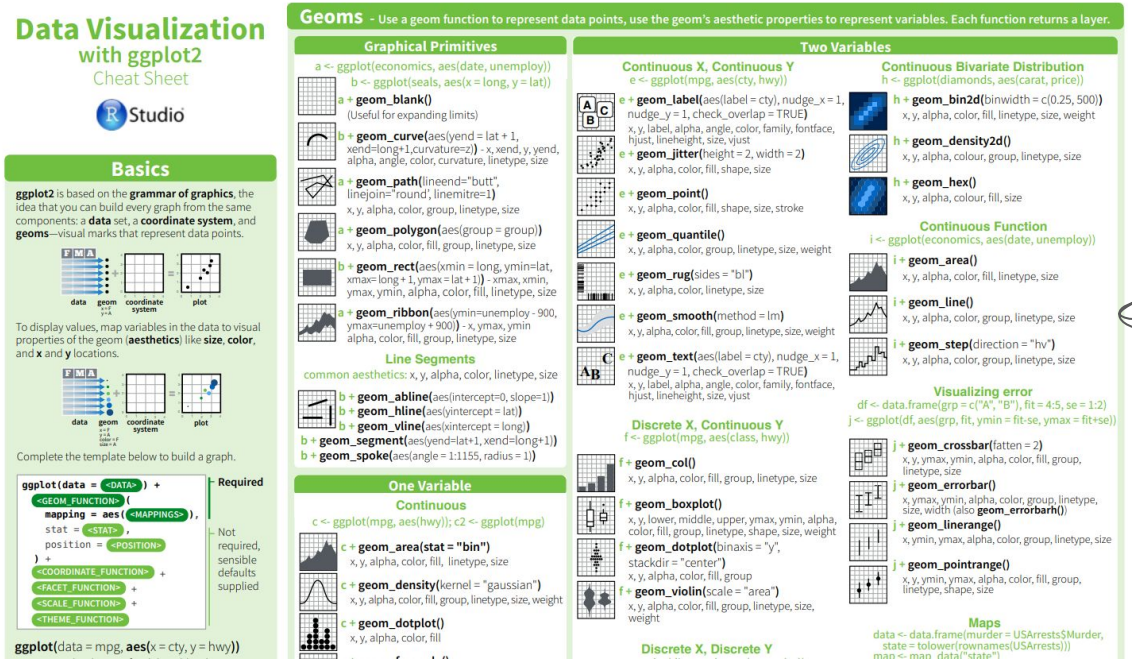
```
library(tidyverse) # include ggplot2

ggplot(data = mtcars, mapping = aes(x = wt, y = mpg)) +
    geom_point()
```

three required elements to make ggplot

We use + to add more layers

ggplot cheat sheet



Data Visualization with ggplot2 Cheat Sheet

RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

data | geom | coordinate system | plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data | geom | coordinate system | plot

Complete the template below to build a graph.

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
    mapping = aes(<MAPPINGS>),
    stat = <STAT>,
    position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

Required

Not required, sensible defaults supplied

```
ggplot(data = mpg, aes(x = cty, y = hwy))
```

Geoms - Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

Graphical Primitives

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + **geom_blank()**
(Useful for expanding limits)

b + **geom_curve**(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + **geom_path**(lineend="butt", linejoin="round', linemitre=1)
x, y, alpha, color, group, linetype, size

a + **geom_polygon**(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + **geom_rect**(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + **geom_ribbon**(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin
alpha, color, fill, group, linetype, size

Line Segments

common aesthetics: x, y, alpha, color, linetype, size

b + **geom_abline**(aes(intercept=0, slope=1))
b + **geom_hline**(aes(yintercept = lat))
b + **geom_vline**(aes(xintercept = long))
b + **geom_segment**(aes(yend=lat+1, xend=long+1))
b + **geom_spoke**(aes(angle = 1:1155, radius = 1))

One Variable

Continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size

c + **geom_density**(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + **geom_dotplot()**
x, y, alpha, color, fill

Two Variables

Continuous X, Continuous Y

e <- ggplot(mpg, aes(cty, hwy))

e + **geom_label**(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + **geom_jitter**(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + **geom_point()**
x, y, alpha, color, fill, shape, size, stroke

e + **geom_quantile()**
x, y, alpha, color, group, linetype, size, weight

e + **geom_rug**(sides = "bl")
x, y, alpha, color, linetype, size

e + **geom_smooth**(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

e + **geom_text**(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

f <- ggplot(mpg, aes(class, hwy))

f + **geom_col()**
x, y, alpha, color, fill, group, linetype, size

f + **geom_boxplot()**
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + **geom_dotplot**(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill, group

f + **geom_violin**(scale = "area")
x, y, alpha, color, fill, group, linetype, size, weight

Discrete X, Discrete Y

Continuous Bivariate Distribution

h <- ggplot(diamonds, aes(carat, price))

h + **geom_bin2d**(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + **geom_density2d()**
x, y, alpha, colour, group, linetype, size

h + **geom_hex()**
x, y, alpha, colour, fill, size

Continuous Function

i <- ggplot(economics, aes(date, unemploy))

i + **geom_area()**
x, y, alpha, color, fill, linetype, size

i + **geom_line()**
x, y, alpha, color, group, linetype, size

i + **geom_step**(direction = "hv")
x, y, alpha, color, group, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + **geom_crossbar**(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + **geom_errorbar()**
x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + **geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size

j + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

Maps

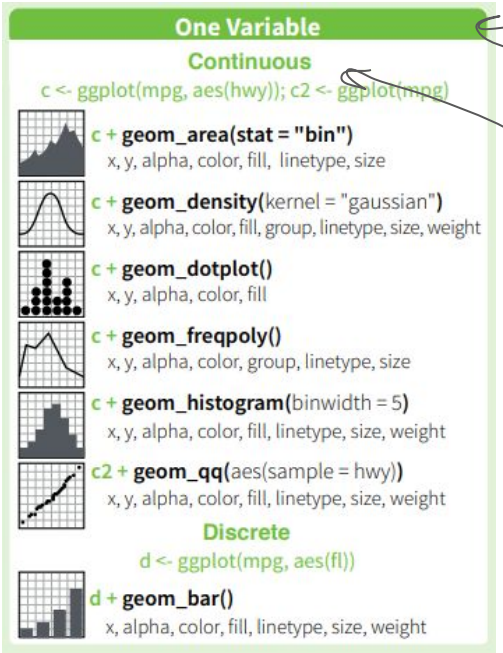
data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")

Free Download

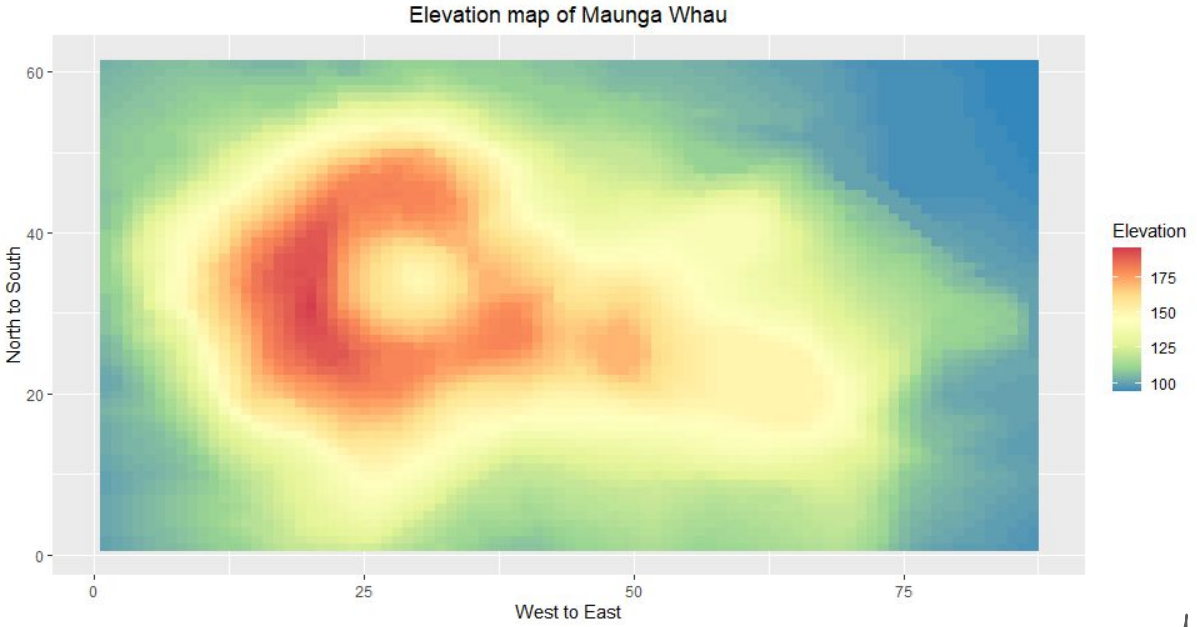
[ggplot2-cheatsheet-2.1 (rstudio.com)]

Rule of thumb

1. How many variables do we have?
2. Data type of our variables?



Volcano Plots (geom_raster)




```
ggplot(volcano, aes(Var1, Var2)) +
  geom_raster(aes(fill=value)) +
  scale_fill_distiller(palette = "Spectral",
direction = -1) +
  labs(x="West to East",
       y="North to South",
       title = "Elevation map of Maunga Whau",
       fill = "Elevation")
```

Dataset

https://gist.githubusercontent.com/toyeiei/118136402ffb5d413004fff1549c5705c/raw/099309cb808b994fa5833680cc243b3246fd3398/volcano_example.csv