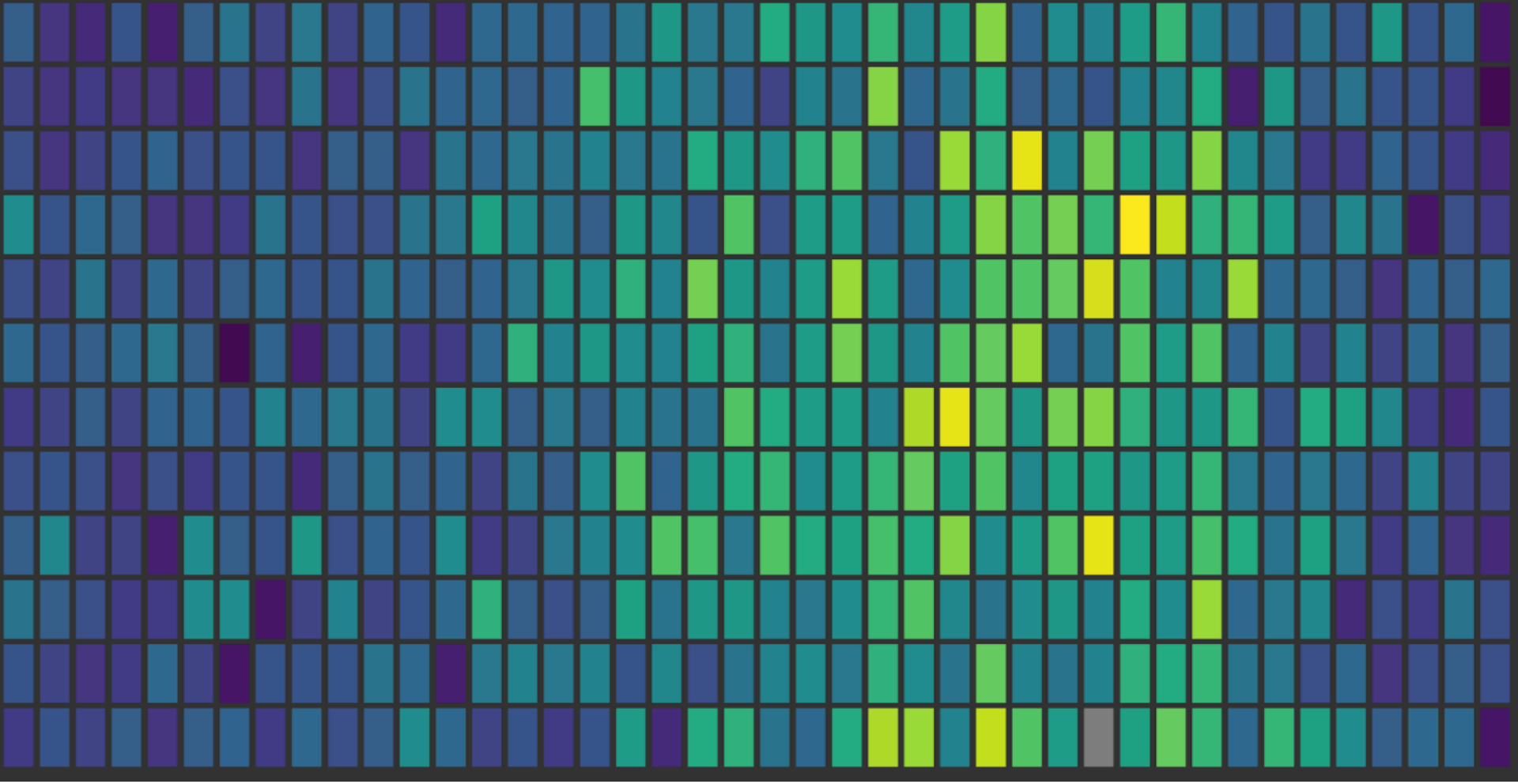


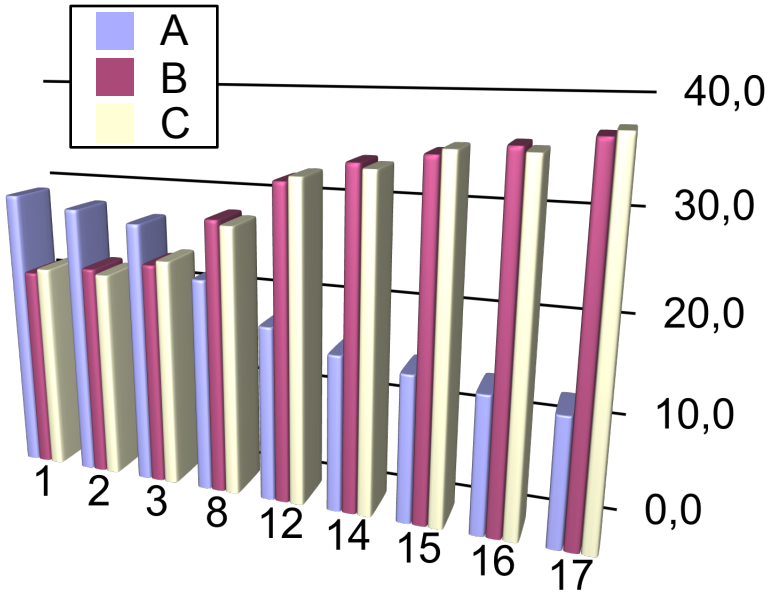
Graphics

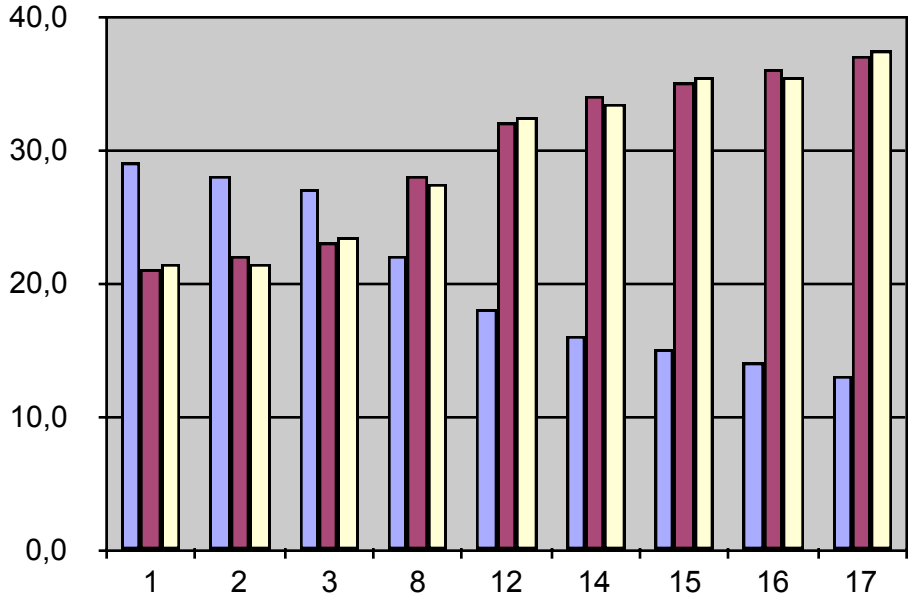
W.Huber - Laura Symul - Susan Holmes Oct 2019

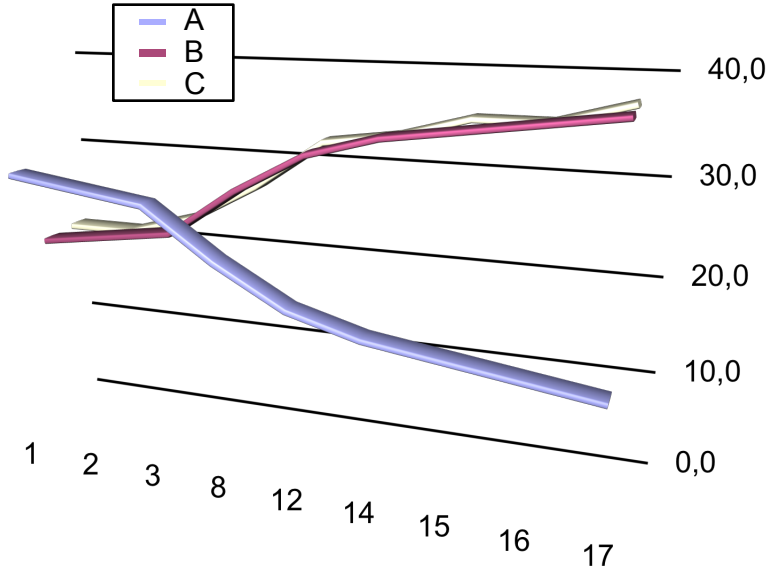
Why graphics?

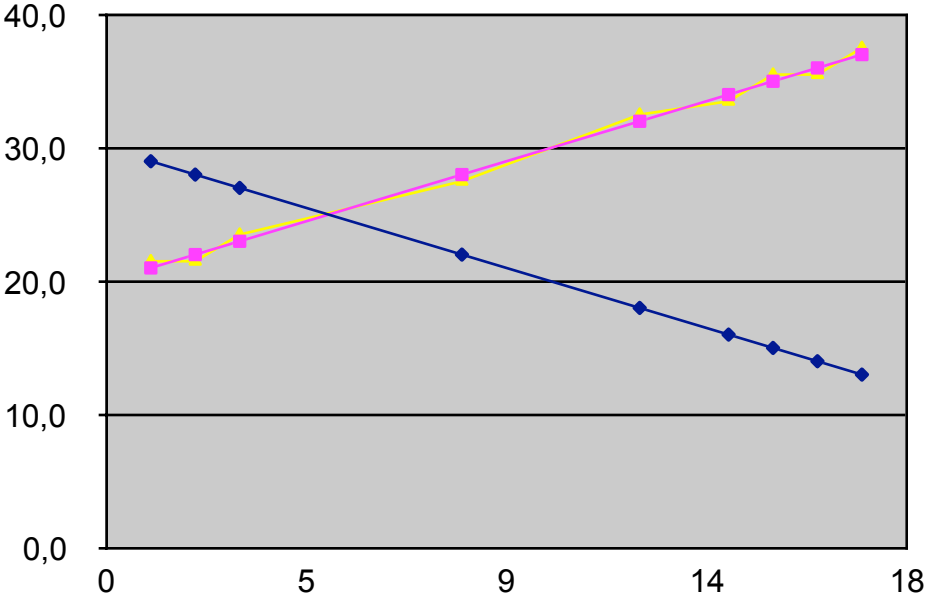
- To explore data
- To communicate data patterns & preliminary insights with collaborators
- To display results and convey findings in a publication


Horror Picture Show


A
B
C
40,0
30,0
20,0
10,0
0,0
1
2
3
8
12
14
15
16
17


40,0
30,0
20,0
10,0
0,0
1
2
3
8
12
14
15
16
17


A
B
C
40,0
30,0
20,0
10,0
0,0
1
2
3
8
12
14
15
16
17

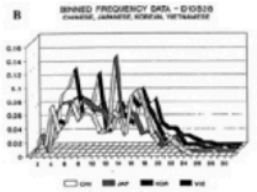

40,0
30,0
20,0
10,0
0,0
0
5
9
14
18

The top ten worst graphs

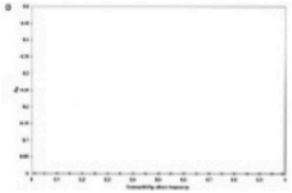
With apologies to the authors, we provide the following list of the top ten worst graphs in the scientific literature. As these examples indicate, good scientists can make mistakes.

---

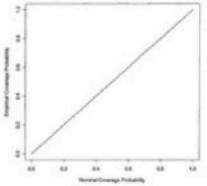
1. Roeder K (1994) DNA fingerprinting: A review of the controversy (with discussion). *Statistical Science* 9:222-278, Figure 4
[The article | The figure | Discussion]



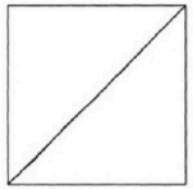
2. Wittke-Thompson JK, Pluzhnikov A, Cox NJ (2005) Rational inferences about departures from Hardy-Weinberg equilibrium. *American Journal of Human Genetics* 76:967-986, Figure 1
[The article | Fig 1AB | Fig 1CD | Discussion]



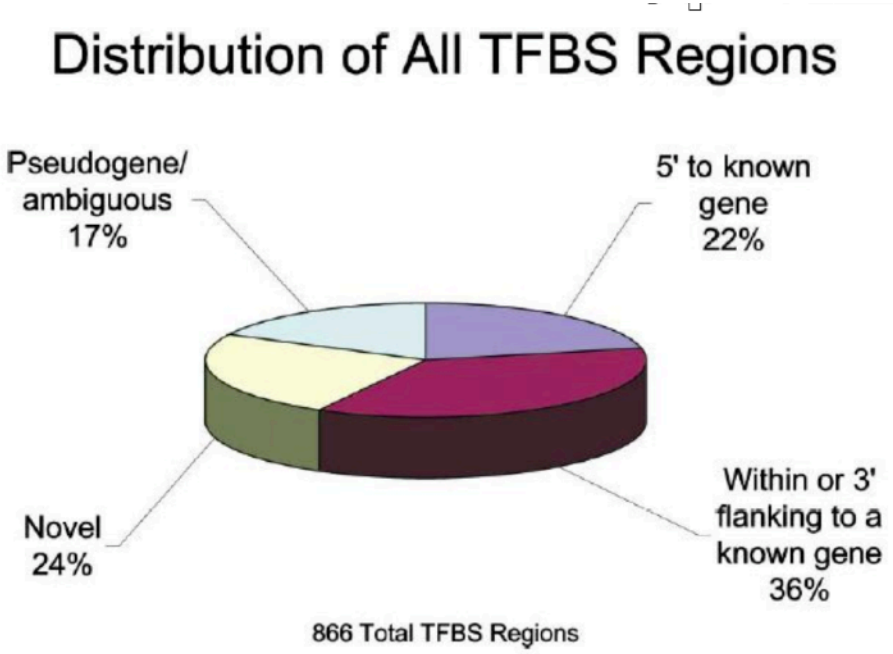
3. Epstein MP, Satten GA (2003) Inference on haplotype effects in case-control studies using unphased genotype data. *American Journal of Human Genetics* 73:1316-1329, Figure 1
[The article | The figure | Discussion]

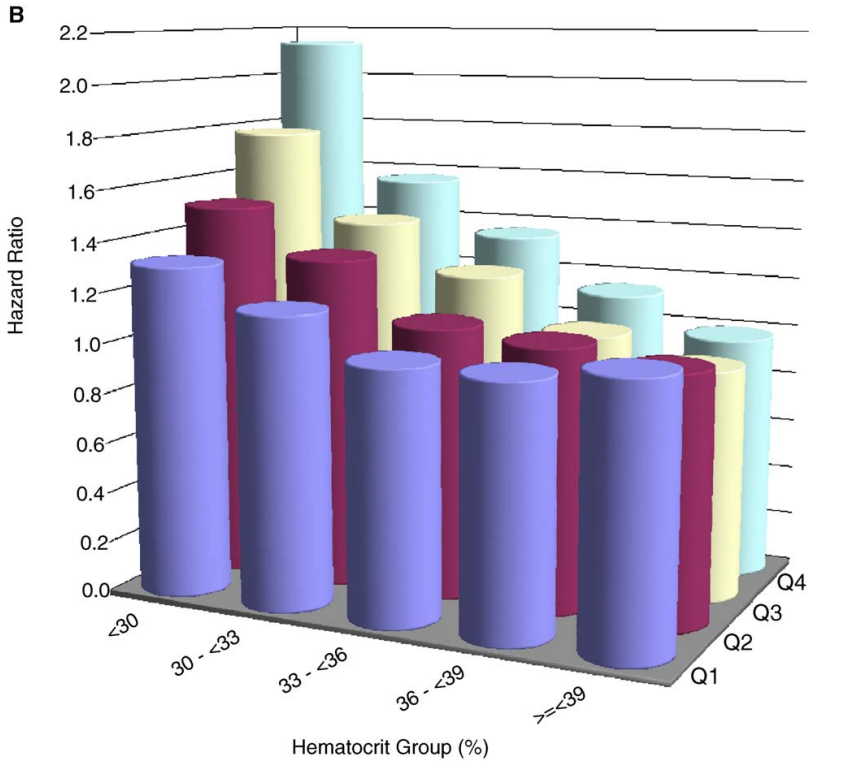


4. Mykland P, Tierney L, Yu B (1995) Regeneration in Markov chain samplers. *Journal of the American Statistical Association* 90:233-241, Figure 1
[The article | The figure | Discussion]



Source: Karl Broman https://www.biostat.wisc.edu/~kbroman/topten_worstgraphs/





Cell
Volume 116, Issue 4, 20 February 2004, Pages 499-509

Article

Unbiased Mapping of Transcription Factor Binding Sites along Human Chromosomes 21 and 22 Points to Widespread Regulation of Noncoding RNAs

Simon Cawley [1, 5], Stefan Bekiranov [1, 5], Huck H Ng [2, 3, 4], Philipp Kapranov [1], Edward A Sekinger [2], Dione Kampa [1], Antonio Piccolboni [1], Victor Sementchenko [1], Jill Cheng [1], Alan J Williams [1], Raymond Wheeler [1], Brant Wong [1], Jorg Drenkow [1], Mark Yamanaka [1], Sandeep Patel [1], Shane Brubaker [1], Hari Tammana [1], Gregg Helt [1] ... Thomas R Gingeras [1]

Show more

ELSEVIER
Journal of Clinical Epidemiology 57 (2004) 1086–1095
Journal of Clinical Epidemiology

Hematocrit was not validated as a surrogate end point for survival among epoetin-treated hemodialysis patients

Dennis J. Cotter[a,*], Kevin Stefanik[a], Yi Zhang[a], Mae Thamer[a], Daniel Scharfstein[b], James Kaufman[c]

[a]Medical Technology and Practice Patterns Institute, Inc., 4733 Bethesda Avenue, Suite 510, Bethesda, MD 20814
[b]Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, Baltimore, MD, 21205-2179
[c]VA Boston Healthcare System, Jamaica Plain, MA 02130

Accepted 30 April 2004

[1] https://doi.org/10.1016/S0092-8674(04)00127-8
[2] https://doi.org/10.1016/j.jclinepi.2004.05.002

Goals for this lecture

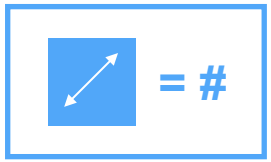
1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

Respect **Graphical Integrity** principles

Respect **Graphical Integrity** principles

Representation of numbers should match the true proportions

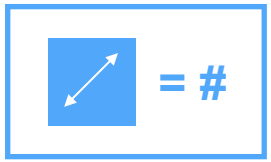
Visual Display of Quantitative Information
E. Tufte

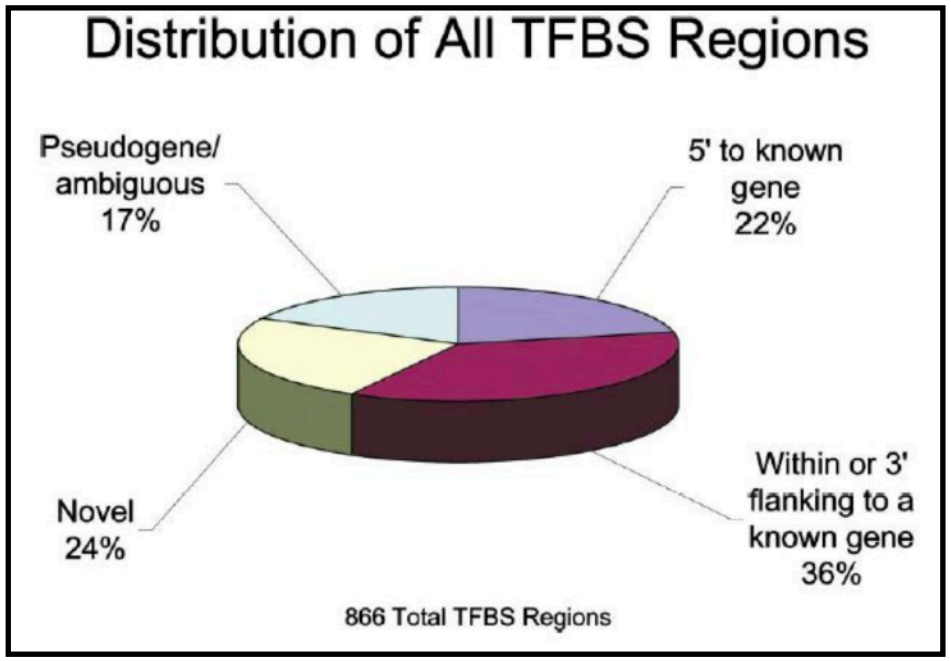


Respect **Graphical Integrity** principles

Representation of numbers should match the true proportions

Visual Display of Quantitative Information
E. Tufte



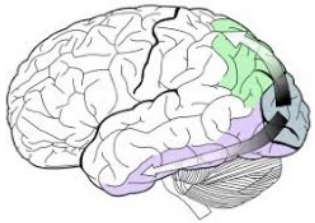


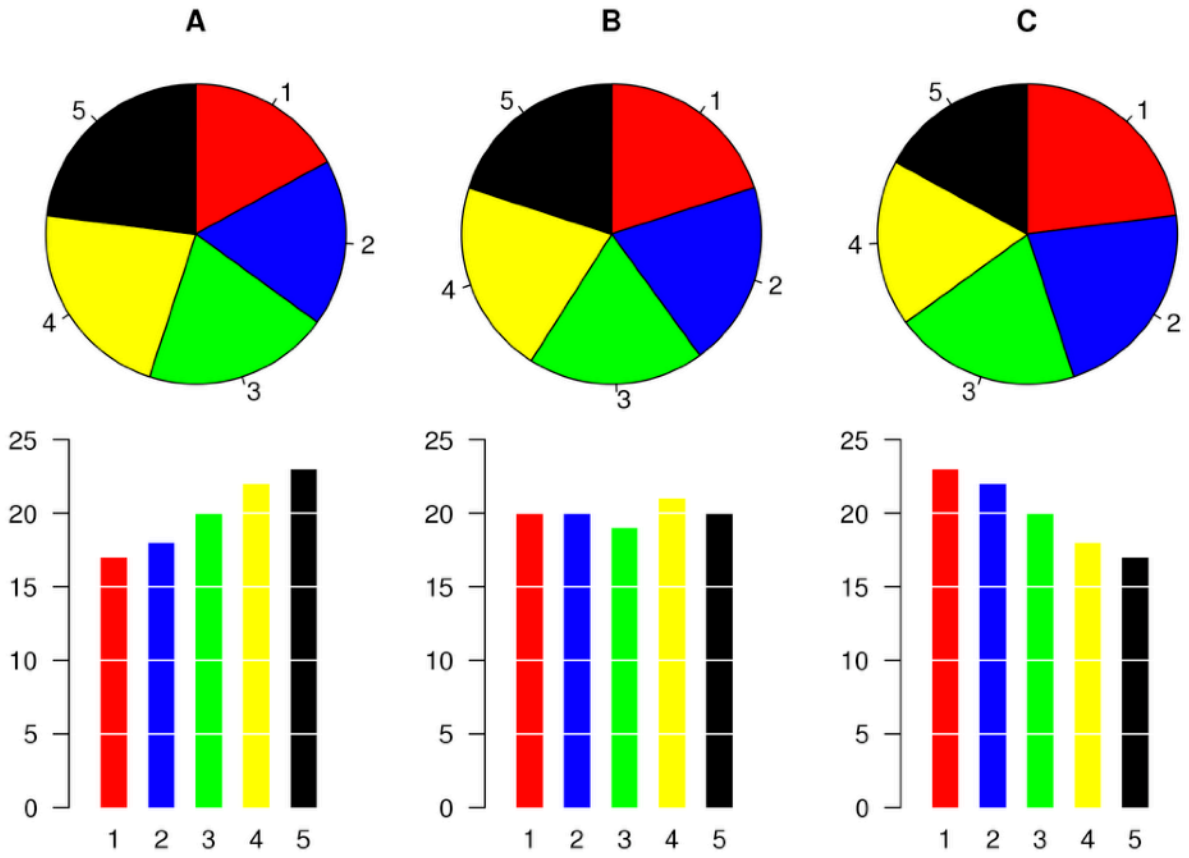
The problem with 3D

The SPACE (or even the actual angles) occupied by each category on the plot is NOT proportional to the actual numbers

This principle also applies to graphics where the origin (0) is not included.

Respect humans' visual abilities

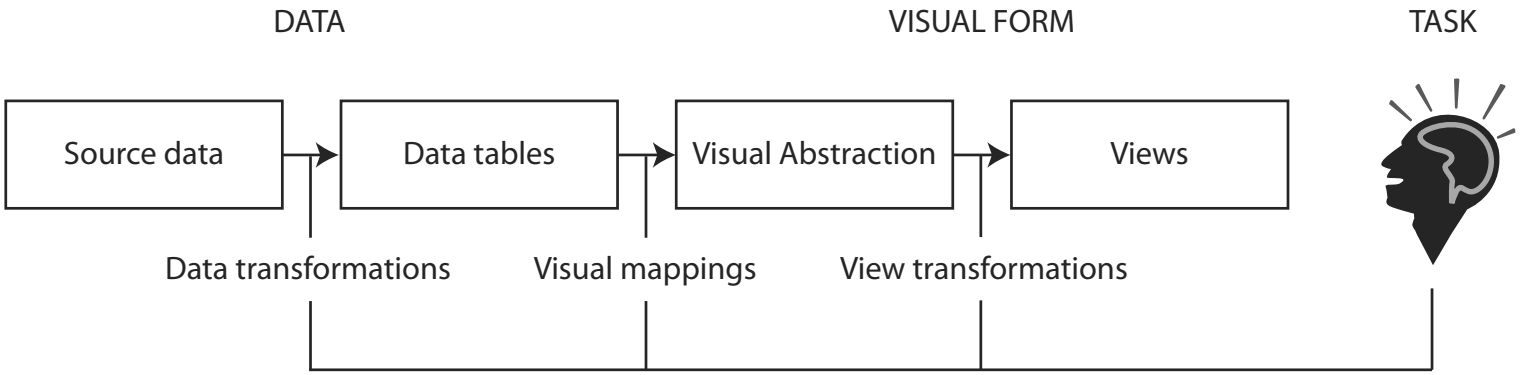




Pie charts are bad because the human brain is not good at differentiating angles. (Especially angles that do not have a horizontal or vertical edge)

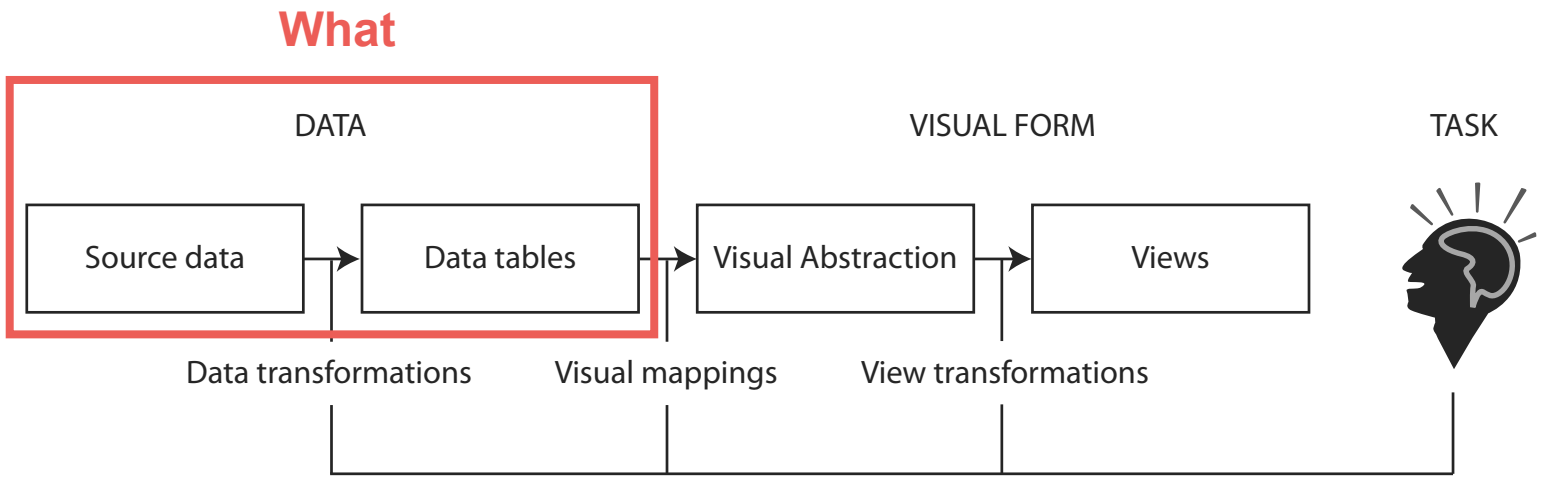
https://medium.com/@kennelliott/39-studies-about-human-perception-in-30-minutes-4728f9e31a73

Choose a plot type that supports the task



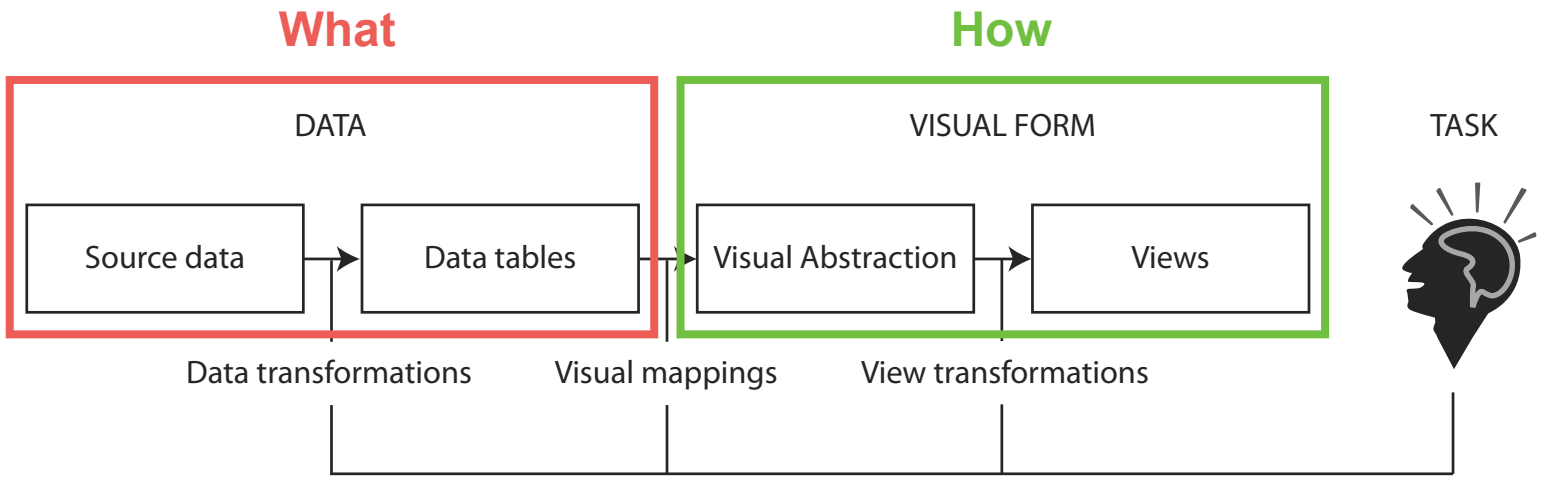
Redrawn from Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman (Editors) (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1558605339.

Choose a plot type that supports the task



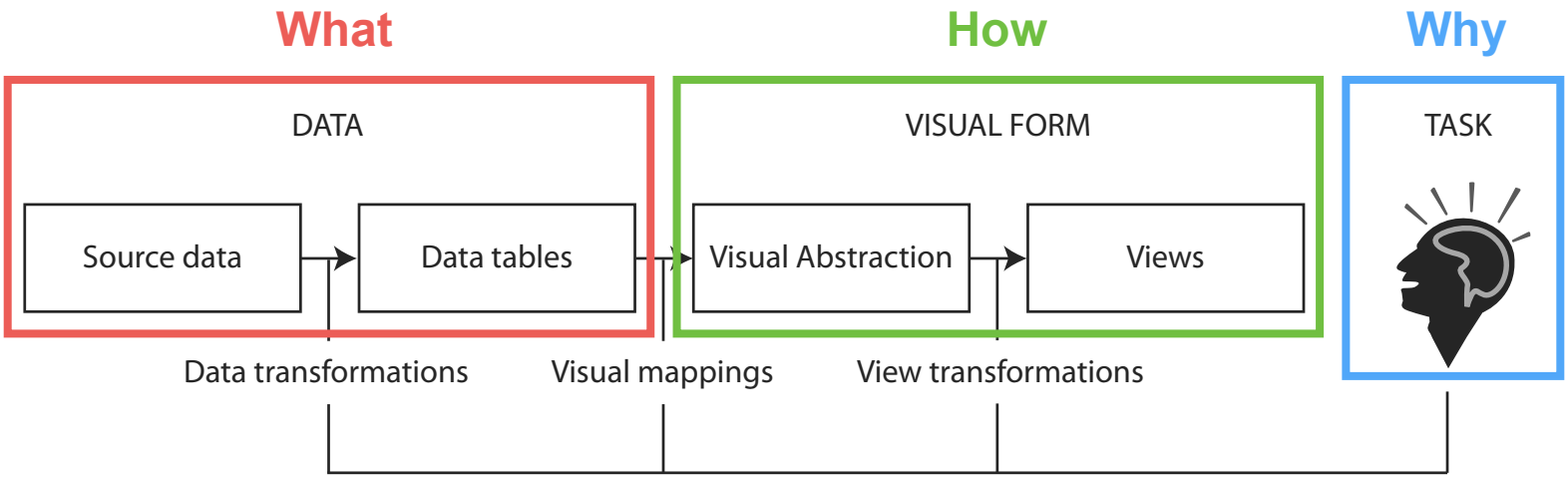
Redrawn from Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman (Editors) (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1558605339.

Choose a plot type that supports the task



Redrawn from Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman (Editors) (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1558605339.

Choose a plot type that supports the task



Redrawn from Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman (Editors) (1999). *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1558605339.

Goals for this lecture

1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

base R plotting

canvas model:
a series of instructions that **sequentially** fill the plotting canvas

base R plotting

canvas model:
a series of instructions that **sequentially** fill the plotting canvas

```
head(DNase)
```

```
##   Run   conc density
## 1   1 0.0488   0.017
## 2   1 0.0488   0.018
## 3   1 0.1953   0.121
## 4   1 0.1953   0.124
## 5   1 0.3906   0.206
## 6   1 0.3906   0.215
```

```
plot(DNase$conc, DNase$density)
```

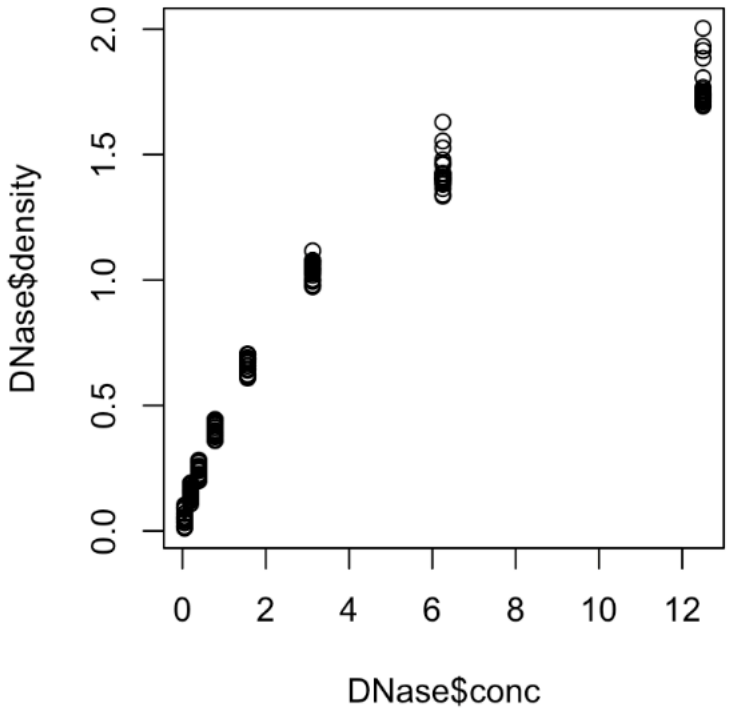


Figure 3.2: Plot of concentration vs. density for an ELISA assay of DNase.

base R plotting

canvas model:
a series of instructions that **sequentially** fill the plotting canvas

Great for quick data exploration!

```
head(DNase)
```

```
##   Run   conc density
## 1   1 0.0488   0.017
## 2   1 0.0488   0.018
## 3   1 0.1953   0.121
## 4   1 0.1953   0.124
## 5   1 0.3906   0.206
## 6   1 0.3906   0.215
```

```
plot(DNase$conc, DNase$density)
```

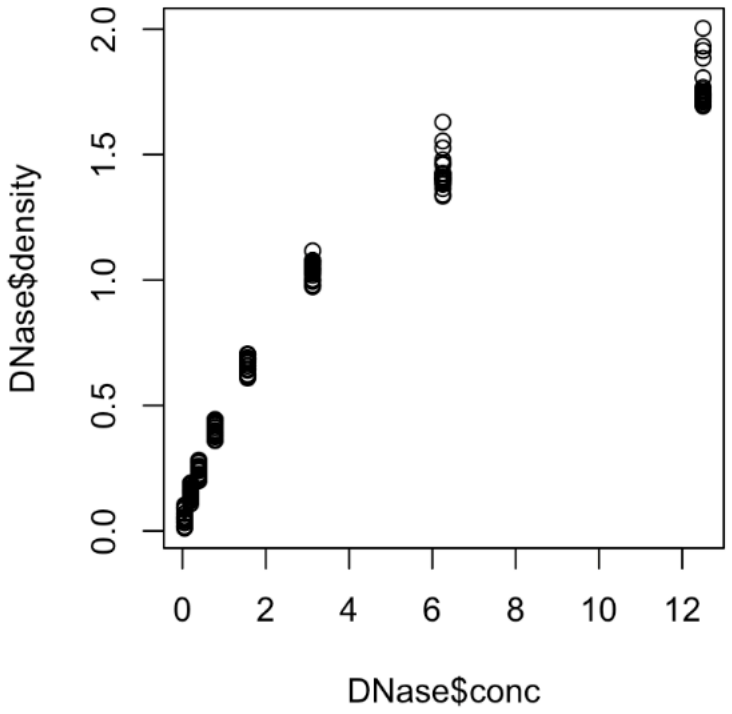
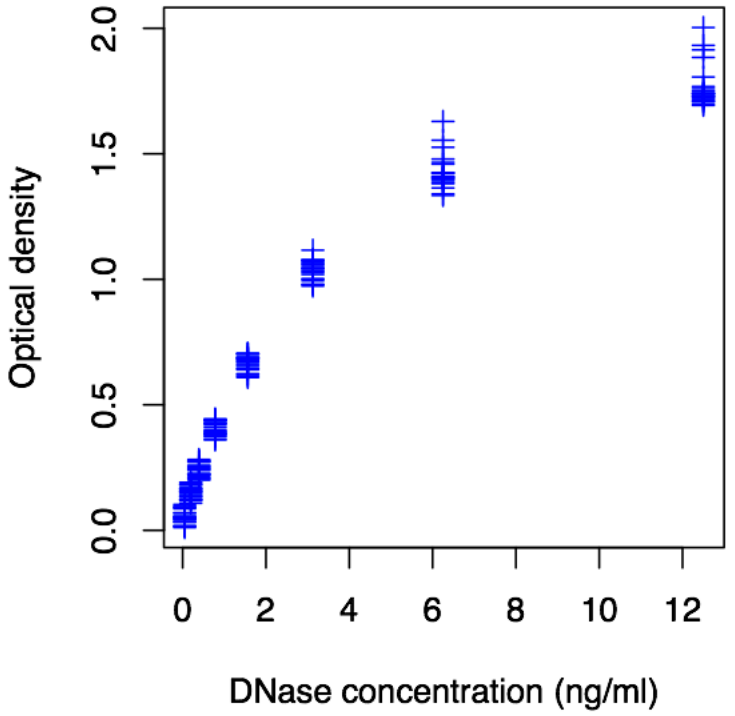


Figure 3.2: Plot of concentration vs. density for an ELISA assay of DNase.

base R plotting

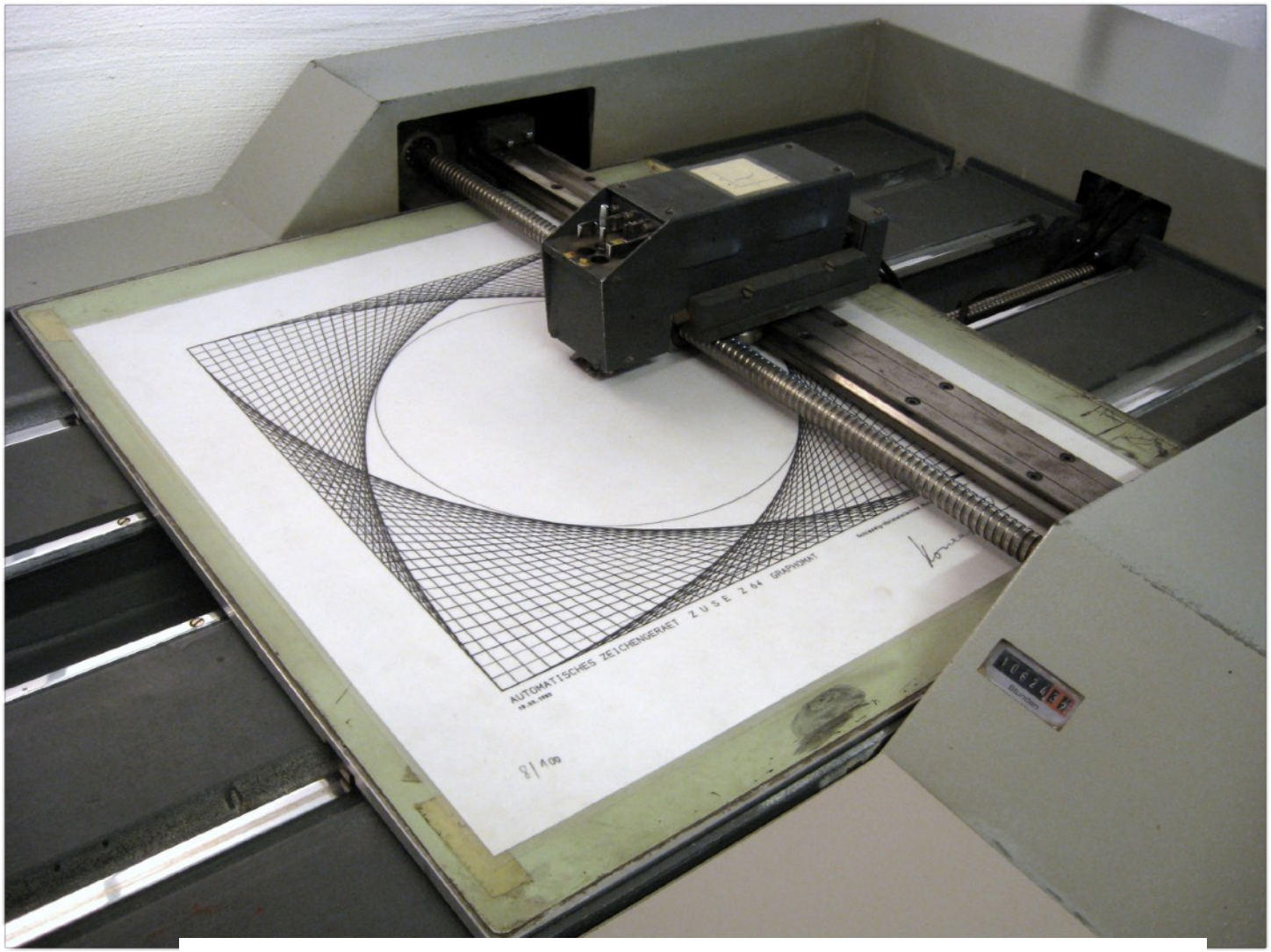
canvas model:
a series of instructions that **sequentially** fill the plotting canvas

Inefficient for customization and generating complex plots.




```
plot(DNase$conc, DNase$density,
ylab = attr(DNase, "labels")$y,
xlab = paste(attr(DNase, "labels")$x, attr(DNase, "units")$x),
pch = 3, col = "blue")
```

base R plotting



ZUSE Plotter Z64 (presented in 1961).

base R plotting

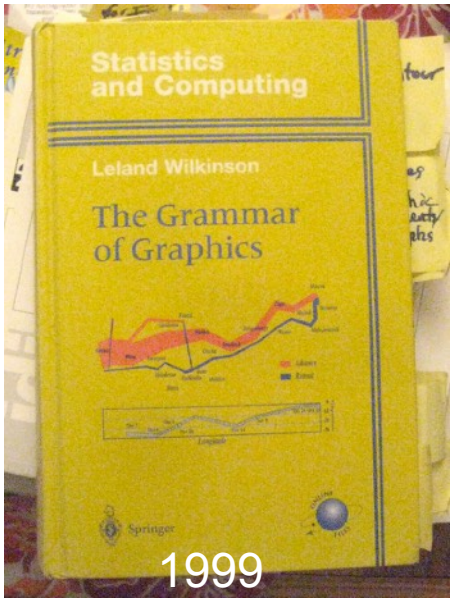
Drawbacks:

- **Layout choices have to be made at the beginning** with no overview over what may still be coming
- **Different functions for different plot types**, with different interfaces
- Routine tasks can require lots of **boilerplate code**
- **No concept of facets / lattices**
- Only a **single global coordinate** system allowed per plot
- **Poor default colours**
- **Resizing** often leads to unsatisfactory results

Goals for this lecture

1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

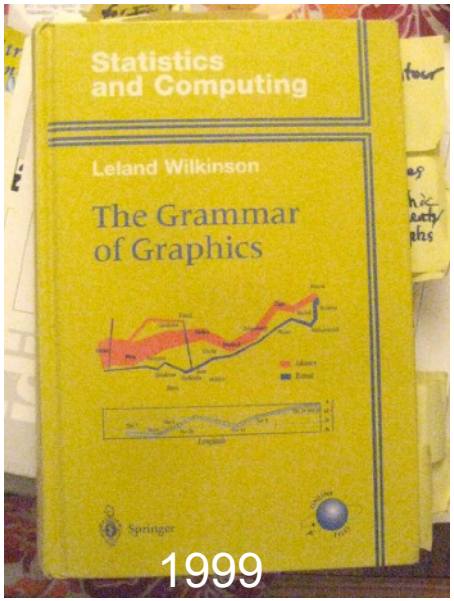
The Grammar of Graphics

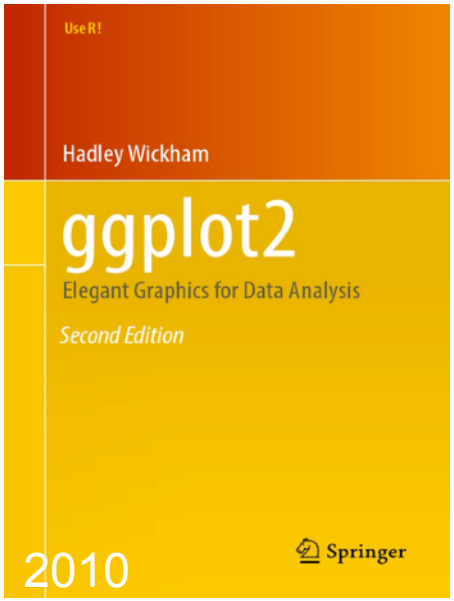


Concept **coined by Leland Wilkinson in 1999**.

An **abstraction** which facilitates reasoning and communicating graphics.

The Grammar of Graphics





Concept **coined by Leland Wilkinson in 1999**.
An **abstraction** which facilitates reasoning and communicating graphics.

ggplot2 is an implementation of **a layered grammar of graphics** that enables users to independently specify the building blocks of a plot and combine them to create just about any kind of graphical display.

`ggplot2` grammar of graphics

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,
- **coordinate systems** and associated **scales** e.g. linear, log, rank,

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,
- **coordinate systems** and associated **scales** e.g. linear, log, rank,
- **facets** separating subsets of data into multiple subplots,

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,
- **coordinate systems** and associated **scales** e.g. linear, log, rank,
- **facets** separating subsets of data into multiple subplots,
- optional parameter settings e.g. text size, font, alignment, legend positions

`ggplot2` grammar of graphics

The components of ggplot2's grammar of graphics are

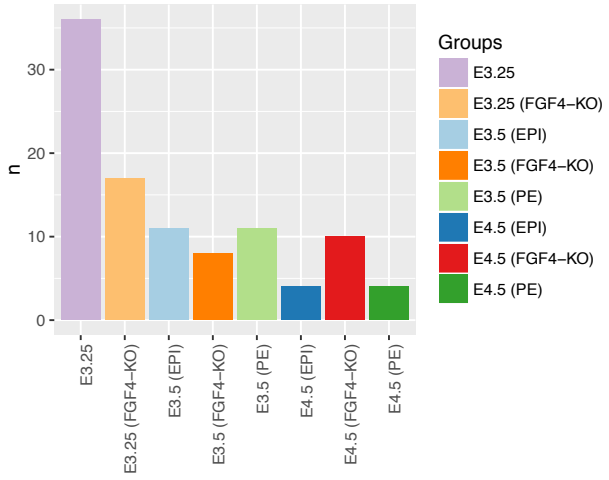
- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,
- **coordinate systems** and associated **scales** e.g. linear, log, rank,
- **facets** separating subsets of data into multiple subplots,
- optional parameter settings e.g. text size, font, alignment, legend positions

ggplot2 grammar of graphics

The components of ggplot2's grammar of graphics are

- **datasets** (*nouns*)
- **geometric objects** (*verbs*), visual representations of the data, e.g. points, lines, rectangles, contours,
- **aesthetics** (*adverbs*), instructions on how to map variables to geometric objects,
- **statistical transformation/summaries** e.g. line fitting, binning,
- **coordinate systems** and associated **scales** e.g. linear, log, rank,
- **facets** separating subsets of data into multiple subplots,
- optional parameter settings e.g. text size, font, alignment, legend positions

```
ggplot(groups, aes(x = sampleGroup, y = n, fill = sampleGroup)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = groupColour, name = "Groups") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



geometric objects

`geom_boxplot()` `stat_boxplot()` — A box and whiskers plot (in the style of Tukey)

`geom_violin()` `stat_ydensity()` — Violin plot

`geom_path()` `geom_line()` `geom_step()` — Connect observations

`geom_point()` — Points

`geom_smooth()` `stat_smooth()` — Smoothed conditional means

`geom_raster()` `geom_rect()` `geom_tile()` — Rectangles

`geom_density()` `stat_density()` — Smoothed density estimates

`geom_density_2d()` `stat_density_2d()` — Contours of a 2d density estimate

Cheat sheet: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

ggplot() template

```
ggplot(data = <default data set>,
       aes(x = <default x axis variable>,
           y = <default y axis variable>,
           ... <other default aesthetic mappings>),
       ... <other plot defaults>) +

  geom_<geom type>(aes(size = <size variable for this geom>,
                       ... <other aesthetic mappings>),
                   data = <data for this point geom>,
                   stat = <statistic string or function>,
                   position = <position string or function>,
                   color = <"fixed color specification">,
                  ... <other arguments, possibly passed to the _stat_ function) +

  scale_<aesthetic>_<type>(name = <"scale label">,
                           breaks = <where to put tick marks>,
                           labels = <labels for tick marks>,
                           ... <other options for the scale>) +

  theme(plot.background = element_rect(fill = "gray"),
        ... <other theme elements>)
```

Data must be in *dataframe* format

```
library(Hiiragi2013)
data(x)
expression <- Biobase::exprs(x)
dftx <- data.frame(pData(x), t(expression))
head(pData(x))
```

```
##         File.name Embryonic.day Total.number.of.cells lineage genotype
## 1 E3.25  1_C32_IN         E3.25                    32                WT
## 2 E3.25  2_C32_IN         E3.25                    32                WT
## 3 E3.25  3_C32_IN         E3.25                    32                WT
## 4 E3.25  4_C32_IN         E3.25                    32                WT
## 5 E3.25  5_C32_IN         E3.25                    32                WT
## 6 E3.25  6_C32_IN         E3.25                    32                WT
##           ScanDate sampleGroup sampleColour
## 1 E3.25 2011-03-16       E3.25      #CAB2D6
## 2 E3.25 2011-03-16       E3.25      #CAB2D6
## 3 E3.25 2011-03-16       E3.25      #CAB2D6
## 4 E3.25 2011-03-16       E3.25      #CAB2D6
## 5 E3.25 2011-03-16       E3.25      #CAB2D6
## 6 E3.25 2011-03-16       E3.25      #CAB2D6
```

```
dim(expression)
```

```
## [1] 45101   101
```

`ggplot()` requires input data in form of a dataframe

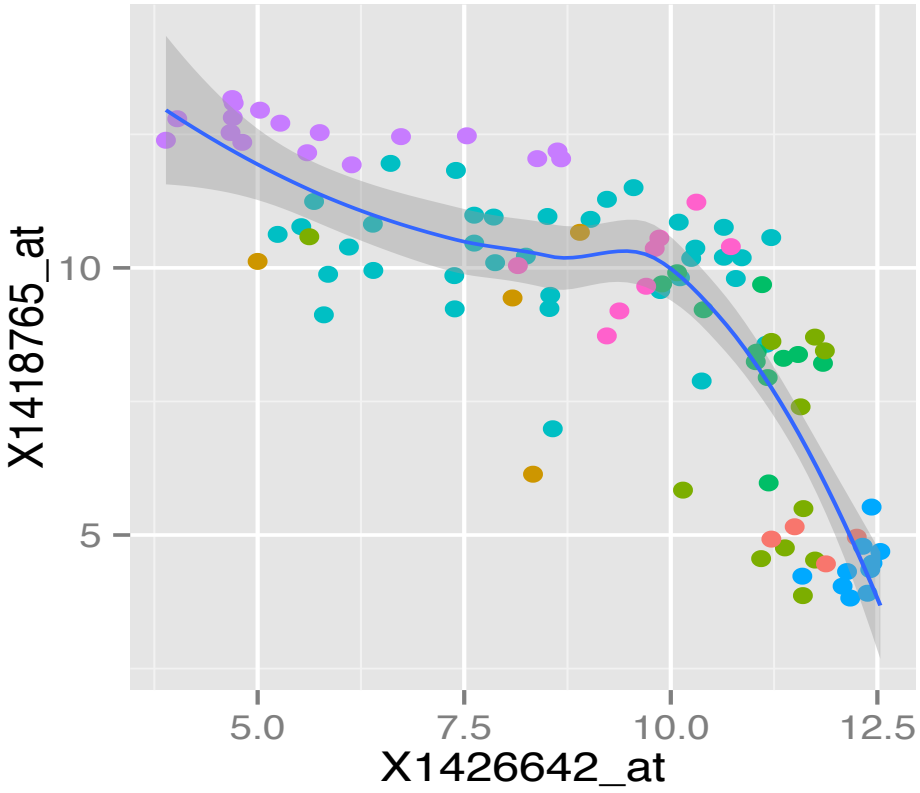
Gene expression **microarray dataset on early development of mouse embryos**

transcriptomes of ~100 individual cells at different time points in. [1]

[1] Cell-to-cell expression variability followed by signal reinforcement progressively segregates early mouse lineages by Ohnishi et al., Nature Cell Biology (2014) 16(1): 27-37. doi: 10.1038/ncb2881.

Multiple layers can be superposed

```
ggplot( dftx, aes( x = X1426642_at, y = X1418765_at ))  +
  geom_point( aes( colour = sampleColour), shape = 19 ) +
  geom_smooth( method = "loess" ) +
  scale_colour_discrete( guide = FALSE )
```



Here, the first layers holds the points, the second holds the smoothed average.

Using the same plot, we can easily change the coordinates

```
groupSize <- table(dftx$sampleGroup)
groupSize
```

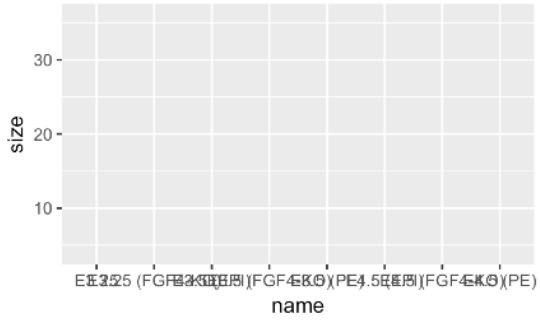
```
pb <- ggplot(data.frame(
                name = names(groupSize),
                size = as.vector(groupSize)),
             aes(x = name, y = size))
```

Using the same plot, we can easily change the coordinates

```
groupSize <- table(dftx$sampleGroup)
groupSize
```

```
pb <- ggplot(data.frame(
                name = names(groupSize),
                size = as.vector(groupSize)),
             aes(x = name, y = size))
```

No geom defined yet!

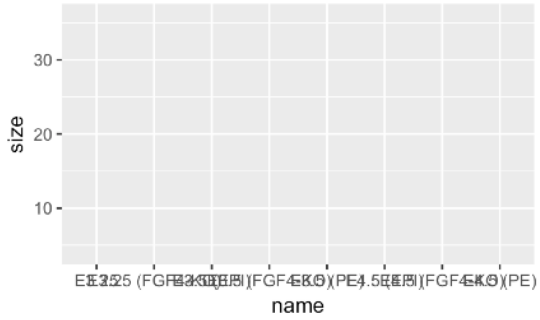


Using the same plot, we can easily change the coordinates

```
groupSize <- table(dftx$sampleGroup)
groupSize
```

```
pb <- ggplot(data.frame(
                  name = names(groupSize),
                  size = as.vector(groupSize)),
               aes(x = name, y = size))
```

No geom defined yet!



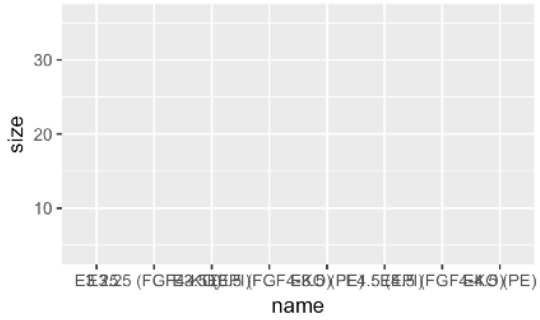
```
pb <- pb + geom_bar(stat = "identity") +
   aes(fill = name) +
   scale_fill_manual(values = groupColour, name = "Colour code") +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))  +
   xlab("Groups") + ylab("Number of Samples")
```

Using the same plot, we can easily change the coordinates

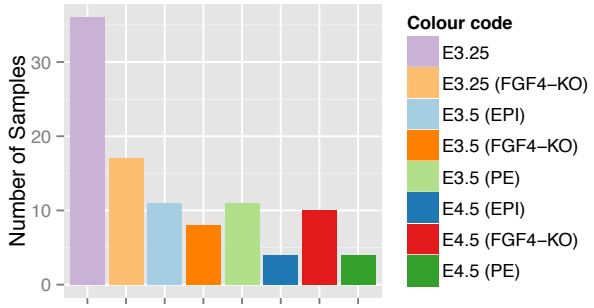
```
groupSize <- table(dftx$sampleGroup)
groupSize
```

```
pb <- ggplot(data.frame(
                  name = names(groupSize),
                  size = as.vector(groupSize)),
               aes(x = name, y = size))
```

No geom defined yet!




```
pb <- pb + geom_bar(stat = "identity") +
   aes(fill = name) +
   scale_fill_manual(values = groupColour, name = "Colour code") +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))  +
   xlab("Groups") + ylab("Number of Samples")
```

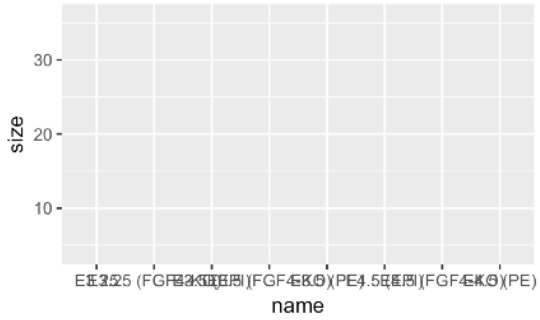


Using the same plot, we can easily change the coordinates

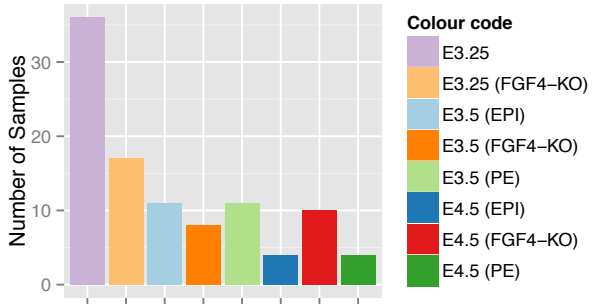
```
groupSize <- table(dftx$sampleGroup)
groupSize
```

```
pb <- ggplot(data.frame(
                  name = names(groupSize),
                  size = as.vector(groupSize)),
               aes(x = name, y = size))
```

No geom defined yet!




```
pb <- pb + geom_bar(stat = "identity") +
   aes(fill = name) +
   scale_fill_manual(values = groupColour, name = "Colour code") +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))  +
   xlab("Groups") + ylab("Number of Samples")
```



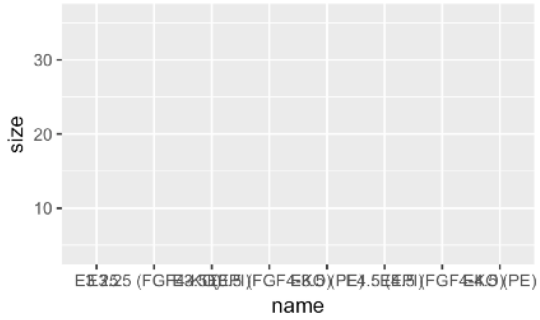

```
pb.polar <- pb + coord_polar() +
  theme(axis.text.x = element_text(angle = 0, hjust = 1),
        axis.text.y = element_blank(),
        axis.ticks = element_blank()) +
  xlab("") + ylab("")
pb.polar
```

Using the same plot, we can easily change the coordinates

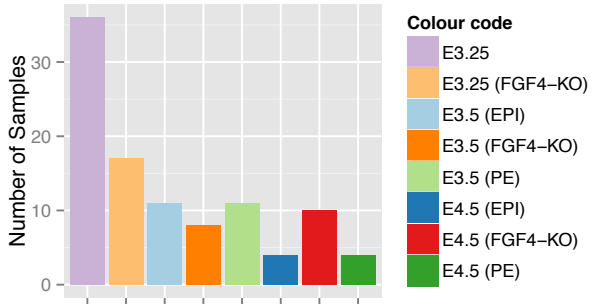
```
groupSize <- table(dftx$sampleGroup)
groupSize
```

```
pb <- ggplot(data.frame(
                  name = names(groupSize),
                  size = as.vector(groupSize)),
               aes(x = name, y = size))
```

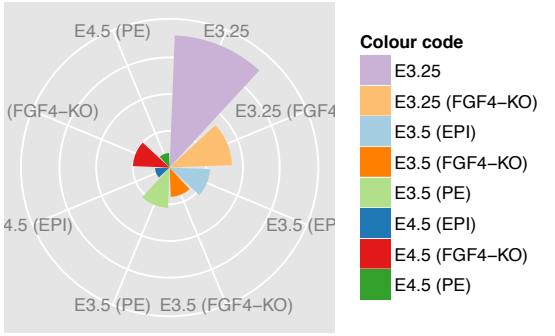
No geom defined yet!




```
pb <- pb + geom_bar(stat = "identity") +
   aes(fill = name) +
   scale_fill_manual(values = groupColour, name = "Colour code") +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))  +
   xlab("Groups") + ylab("Number of Samples")
```



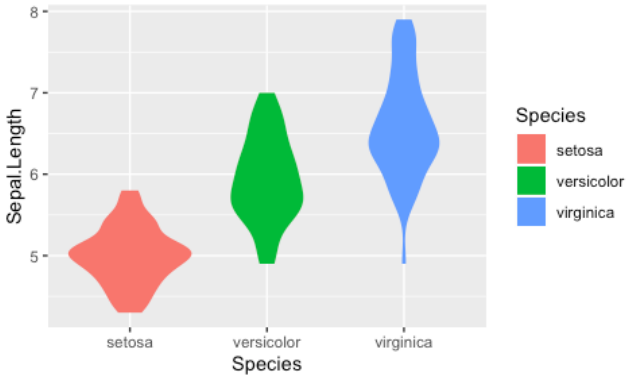

```
pb.polar <- pb + coord_polar() +
  theme(axis.text.x = element_text(angle = 0, hjust = 1),
        axis.text.y = element_blank(),
        axis.ticks = element_blank()) +
  xlab("") + ylab("")
pb.polar
```



Themes can change the look

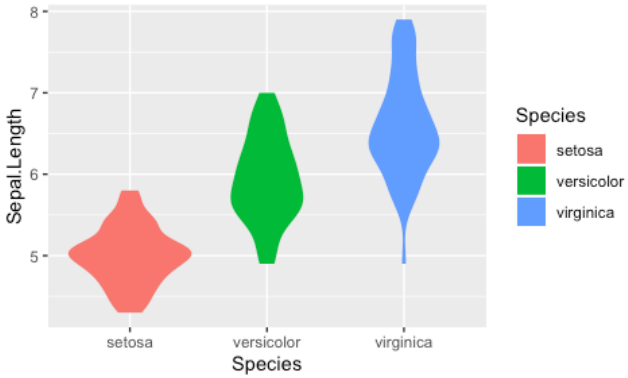
Themes can change the look

```
g = ggplot(iris,
           aes(x = Species,
               y = Sepal.Length,
               fill = Species))+
  geom_violin(col = NA)
g
```

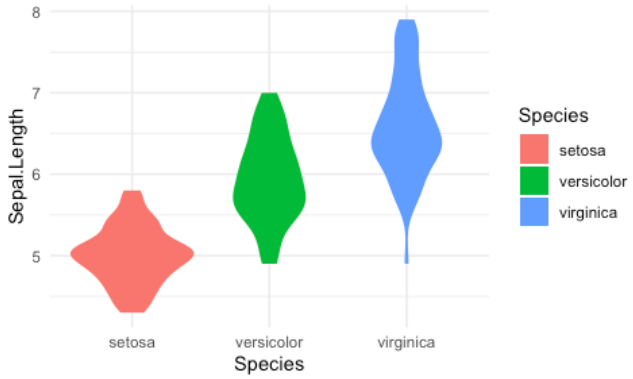


Themes can change the look

```
g = ggplot(iris,
           aes(x = Species,
               y = Sepal.Length,
               fill = Species))+
  geom_violin(col = NA)
g
```

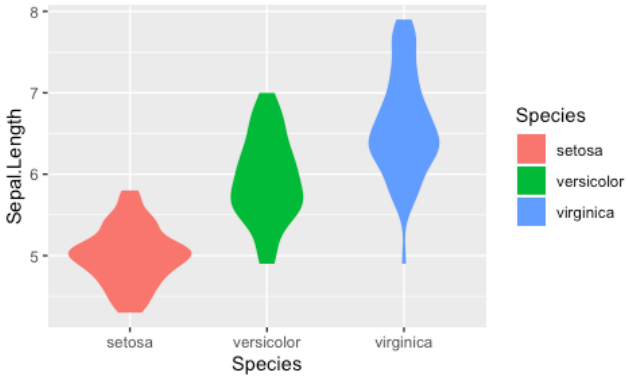



```
g + theme_minimal()
```

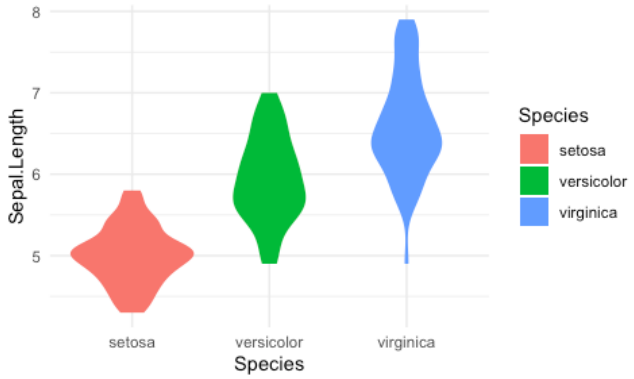


Themes can change the look

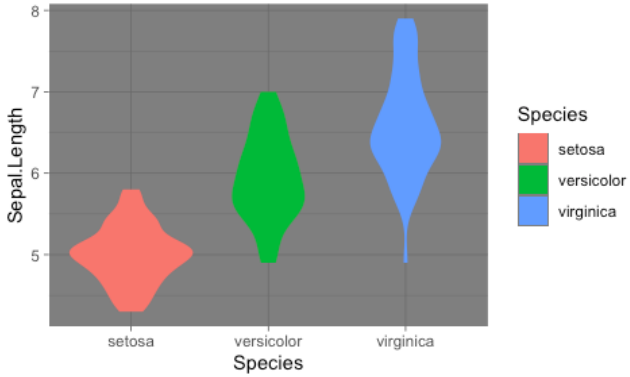
```
g = ggplot(iris,
           aes(x = Species,
               y = Sepal.Length,
               fill = Species))+
  geom_violin(col = NA)
g
```




```
g + theme_minimal()
```

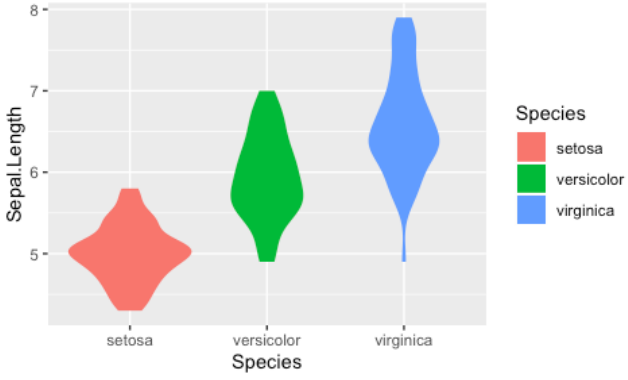



```
g + theme_dark()
```

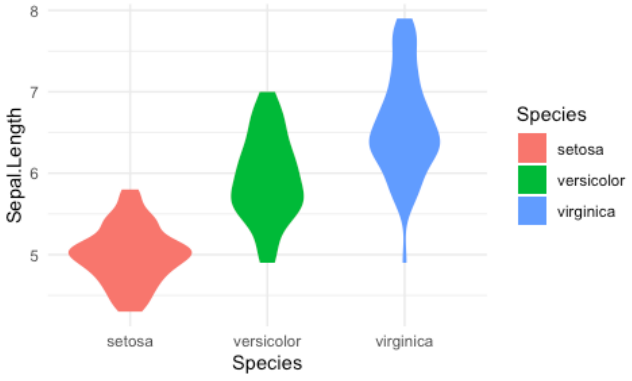


Themes can change the look

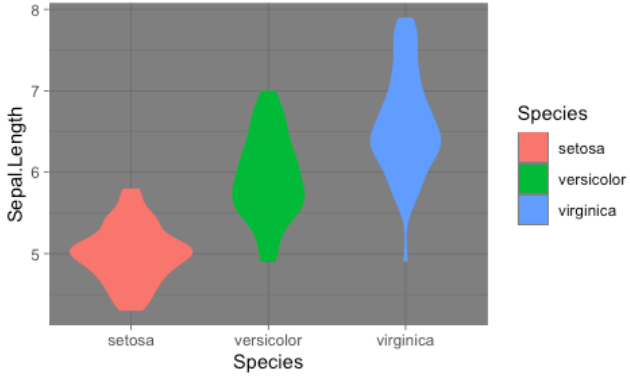
```
g = ggplot(iris,
           aes(x = Species,
               y = Sepal.Length,
               fill = Species))+
  geom_violin(col = NA)
g
```



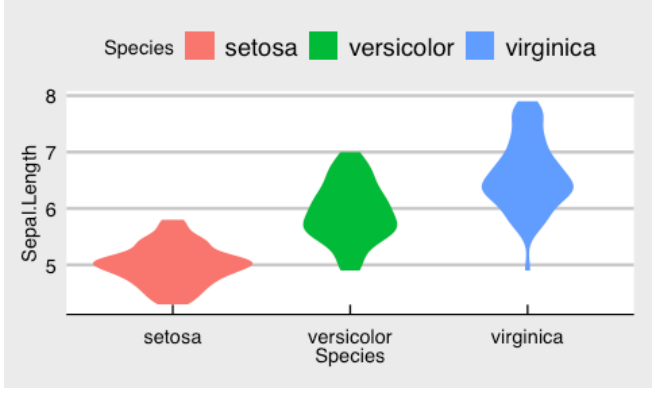

```
g + theme_minimal()
```




```
g + theme_dark()
```




```
library(ggthemes)
g + theme_economist_white()
```



Goals for this lecture

1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

1D plot types

1D plot types

What do you use to show or compare 1D distributions?

1D plot types

What do you use to show or compare 1D distributions?

Boxplot makes sense for **unimodal** distributions

Histogram requires definition of bins/binwidths/break positions. It can create visual artifacts esp. if the number of data points is not large

Density requires setting of **bandwidth parameter**; obscures the sample size (i.e. the uncertainty of the estimate)

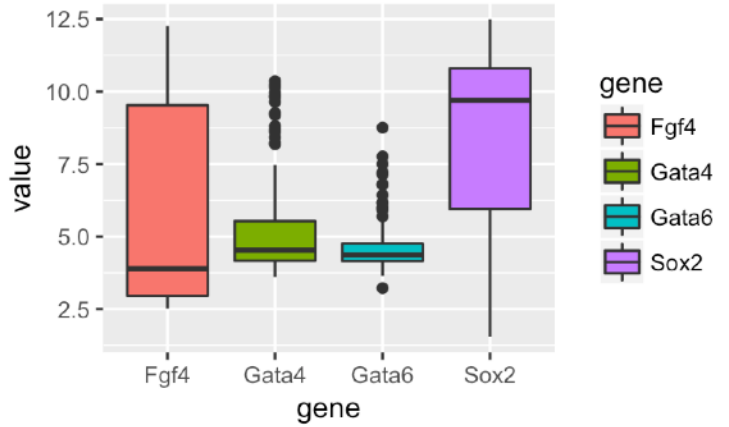
ECDF (Empirical Cumulative Density Function) does not have these problems, but is more abstract and its interpretation requires more training.

If you have only up to a **few dozens of points**
just **show the raw data**! (e.g. with beeswarm)

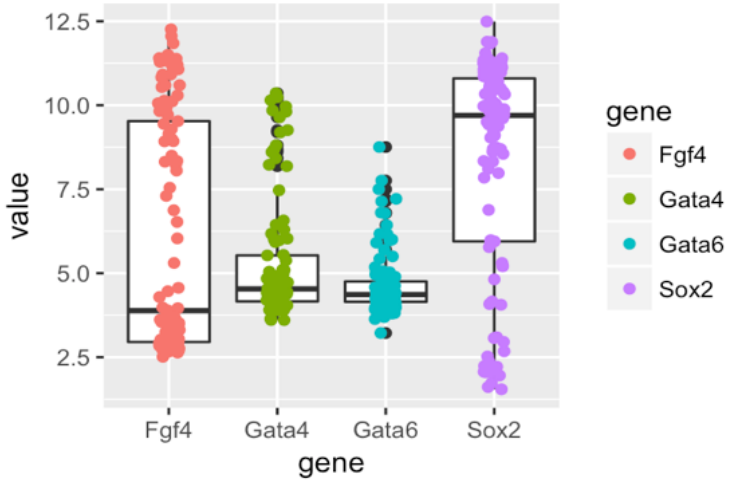
Boxplot

Boxplots are good for plotting summary of 1D continuous data; they allow you to **compare quantiles of data distributions.**

```
p = ggplot(genes, aes( x = gene, y = value))
p + geom_boxplot(aes(fill = gene))
```



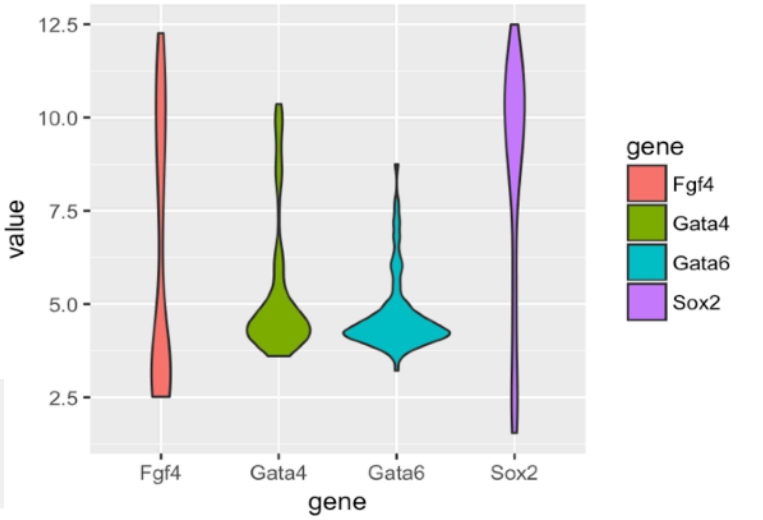

```
p + geom_boxplot() +
  geom_jitter(aes(color = gene), width = 0.1, height = 0)
```



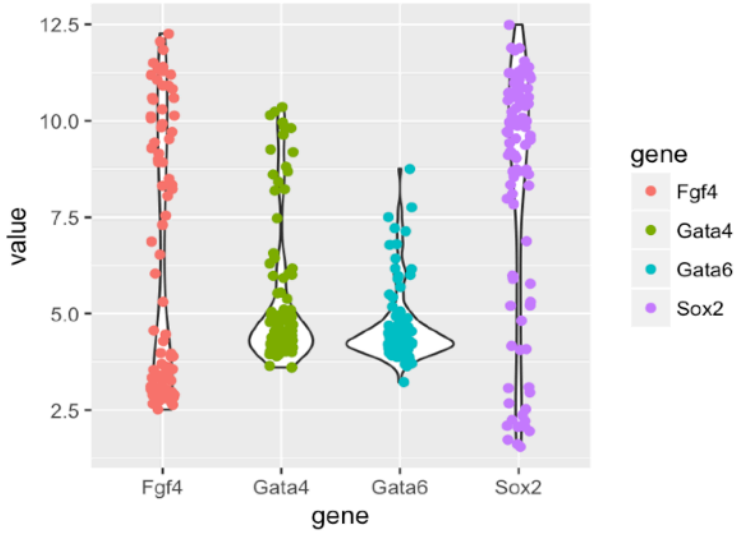
Violin Plot

If there are many observations in the dataset, we can **show the estimated distribution with violin plots.**

```
p = ggplot(genes, aes( x = gene, y = value))
p + geom_violin(aes(fill =  gene))
```

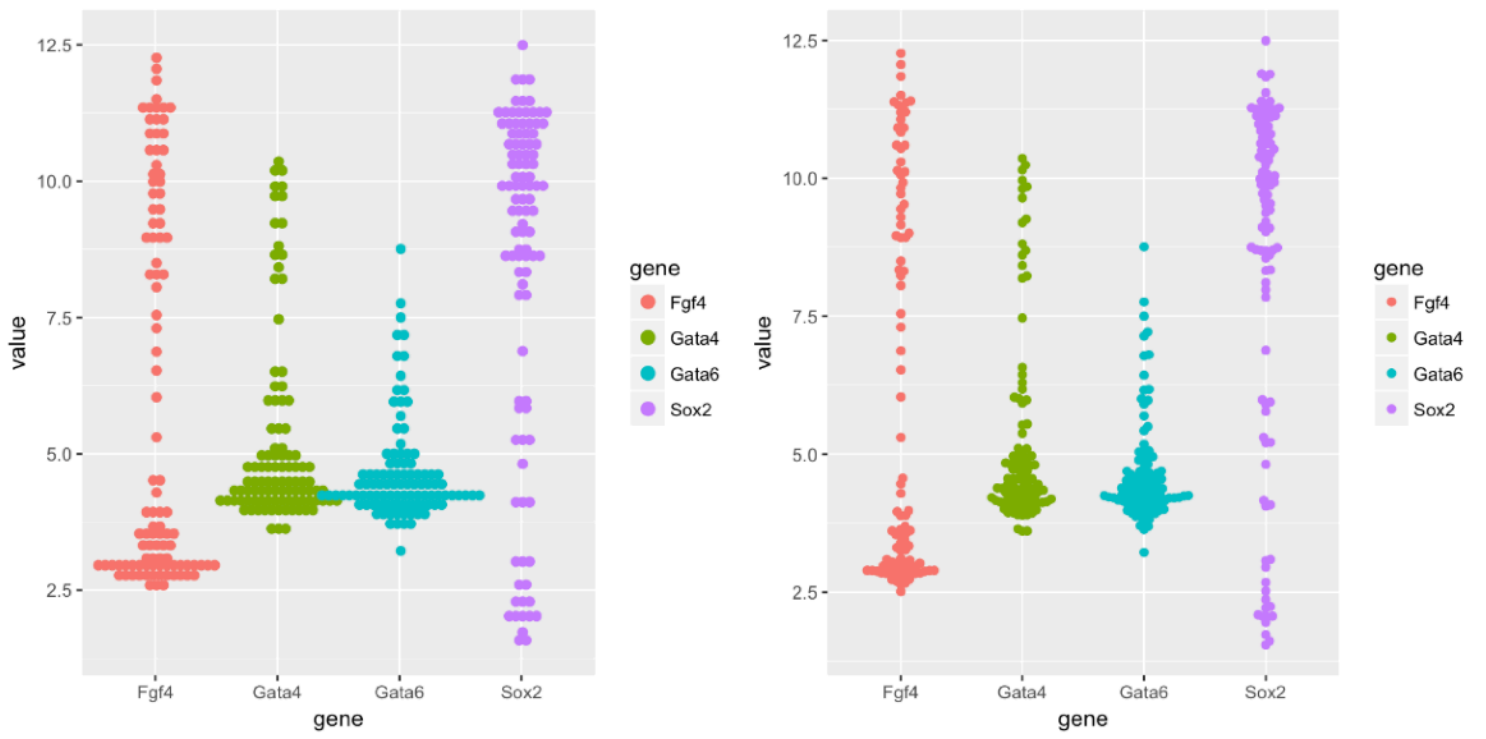



```
p + geom_boxplot() +
  geom_jitter(aes(color = gene), width = 0.1, height = 0)
```



Dot & Beeswarm Plot

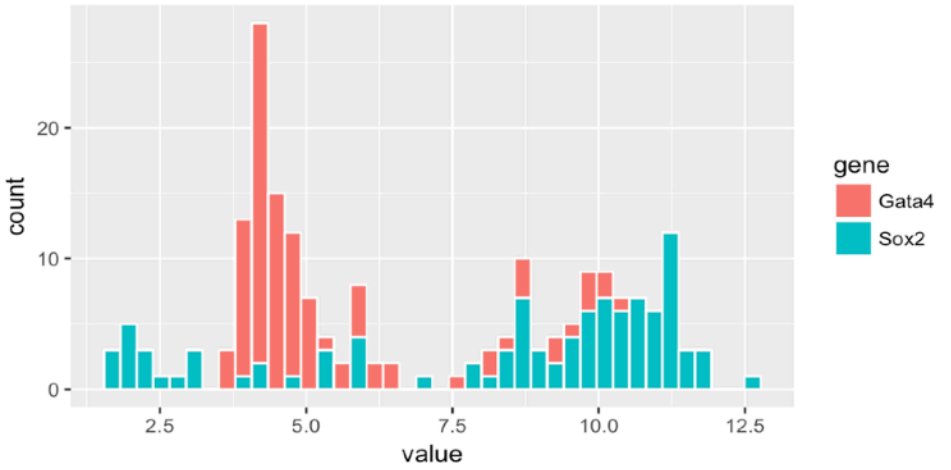
```
p + geom_dotplot(binaxis = "y", binwidth = 1/6,
        stackdir = "center", stackratio = 0.75,
        aes(color = gene))
library("ggbeeswarm")
p + geom_beeswarm(aes(color = gene))
```



Histograms

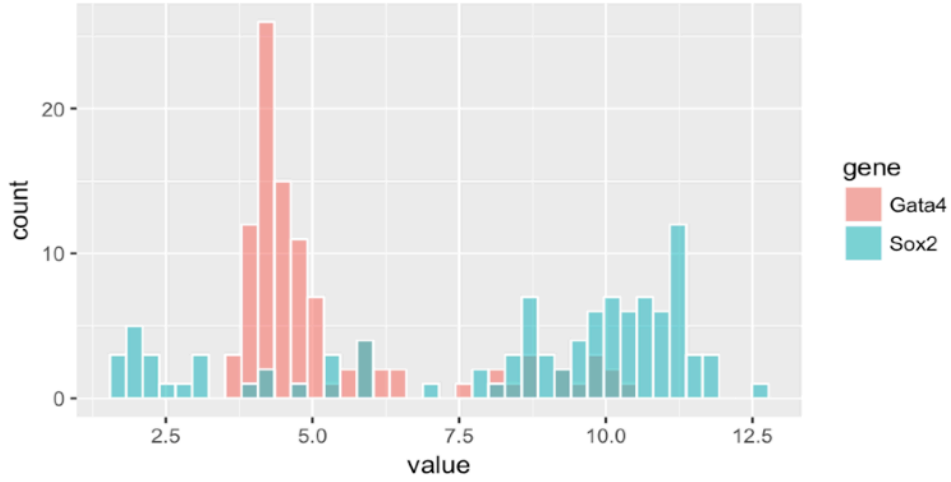
Stacked

```
p = ggplot(genes %>% filter(gene %in% c("Gata4", "Sox2")),
           aes(x = value))
p + geom_histogram(aes(fill = gene),
                   color = "white", bins = 40)
```



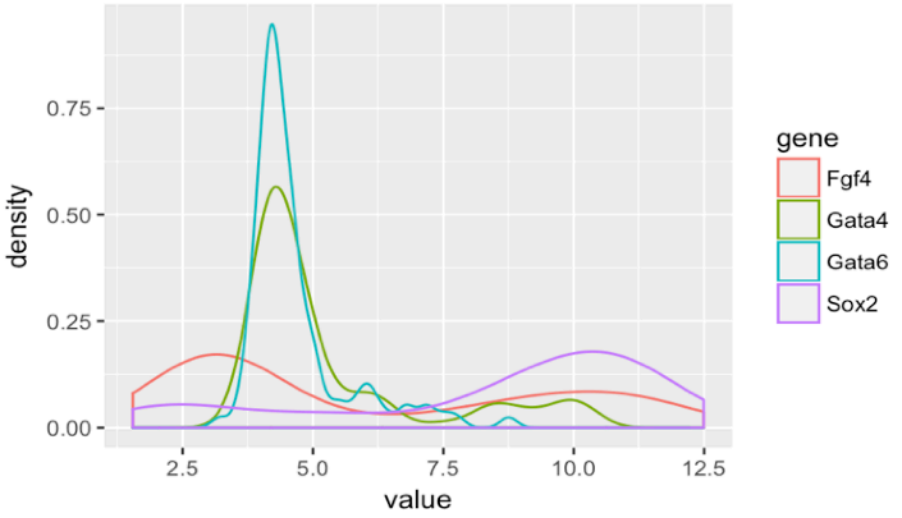
Overlapping

```
p + geom_histogram(
  aes(fill = gene), color="white", alpha=0.6,
  bins = 40, position = "identity")
```

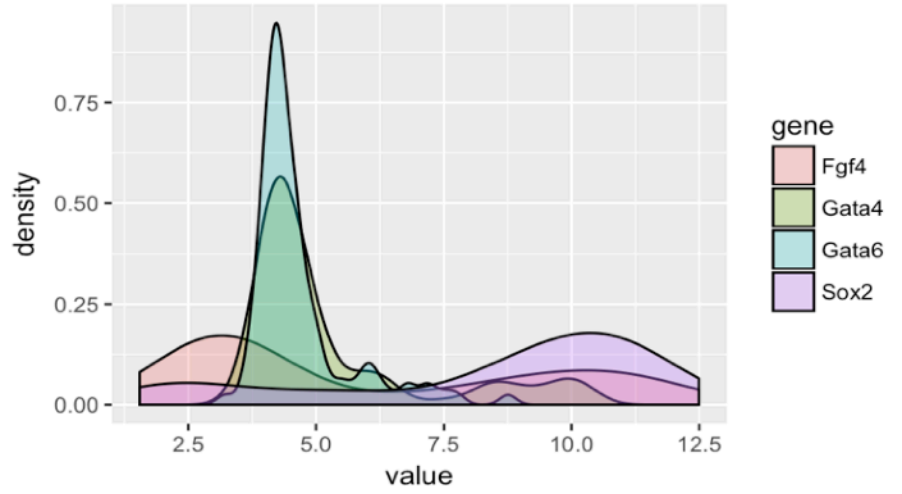


Density plots

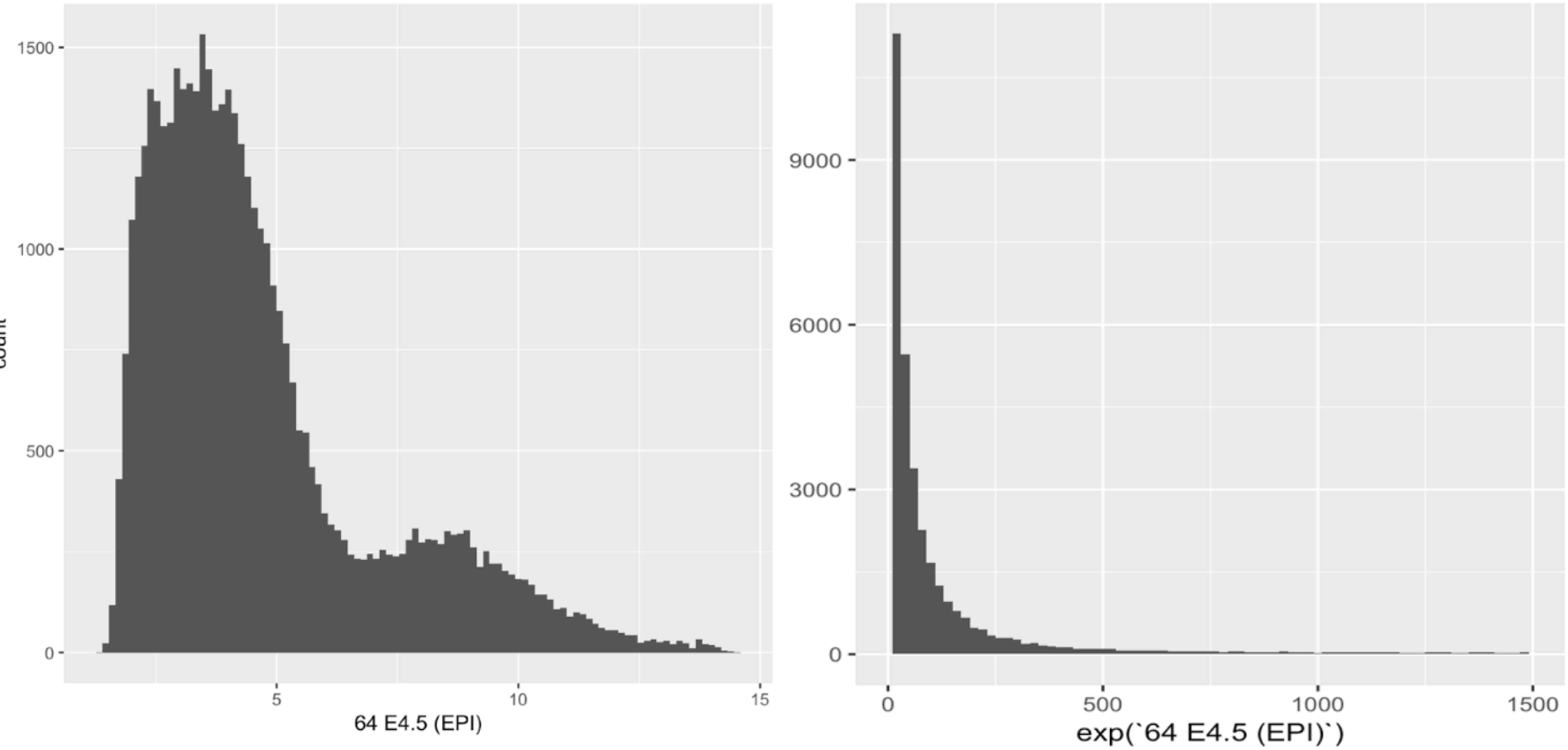
```
p = ggplot(genes, aes( x = value, color = gene))
p + geom_density()
```




```
p = ggplot(genes, aes( x = value, fill = gene))
p + geom_density(alpha = 0.3)
```



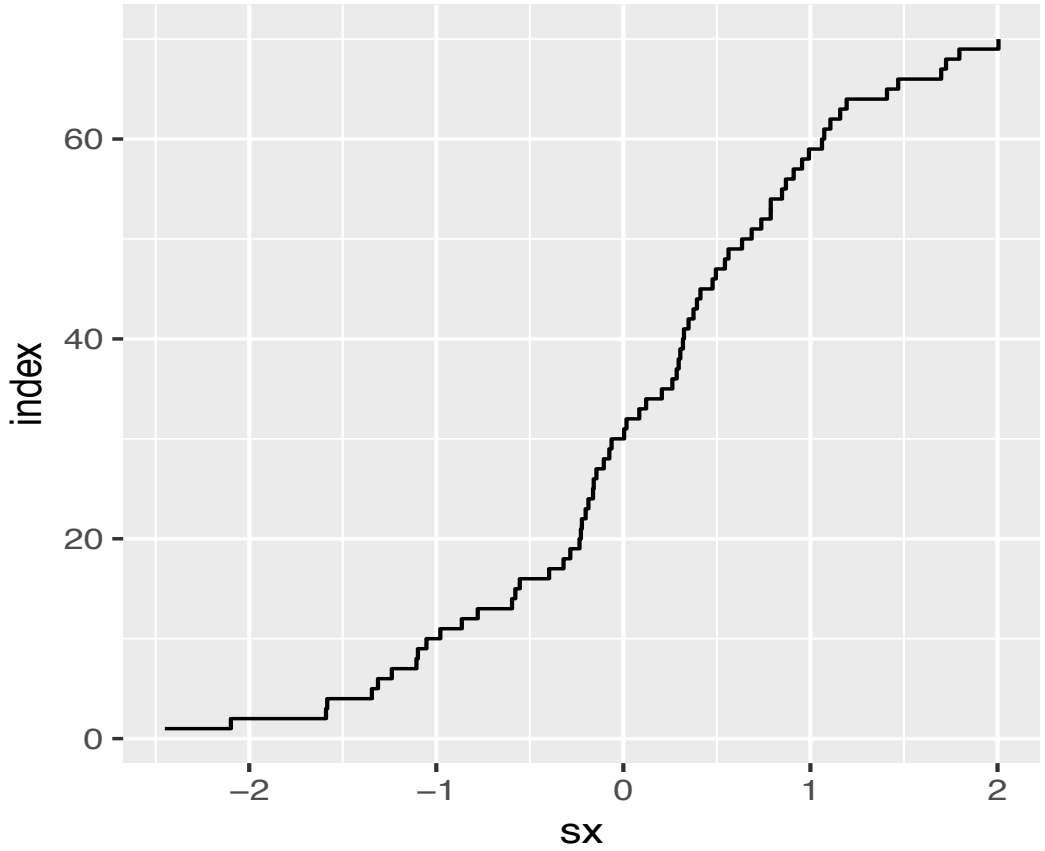
Non-linear transformations change the shape of a density



- The **mode** of a distribution is an **infinitesimal concept**.
- Need either an infinite amount of data or choose smoothing / binning bandwidth
- **Number of modes (let alone their positions) can change under non-linear data transformations**

The empirical cumulative distribution

$$F_n(x) = \frac{\text{number of } i \text{ for which } x_i \leq x}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(x \leq x_i)$$




```
simdata = rnorm(70)
simdf <- data.frame(index = seq(along = simdata), sx = sort(simdata))
ggplot(simdf, aes(x = sx, y = index)) +  geom_step()
```

Bar charts with error bars

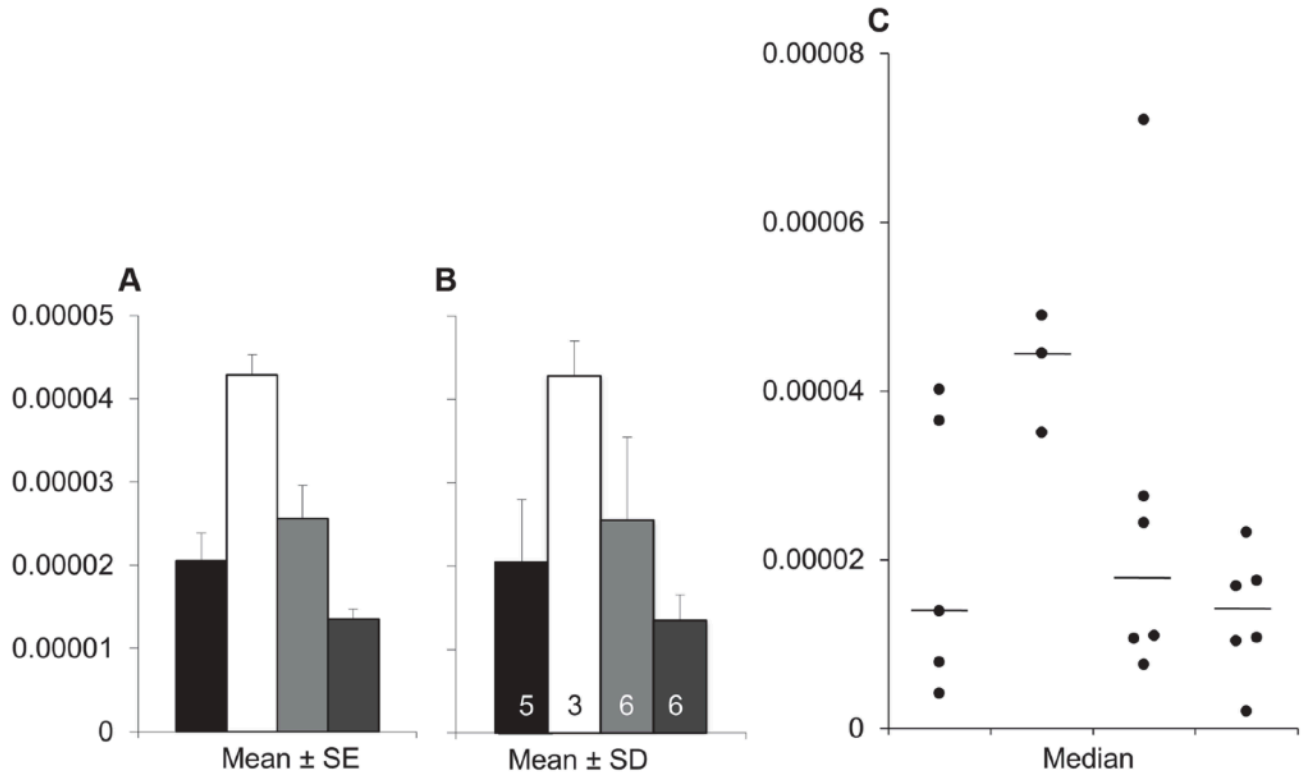


Fig 3. Bar graphs and scatterplots convey very different information. While scatterplots prompt the reader to critically evaluate the statistical tests and the authors' interpretation of the data, bar graphs discourage the reader from thinking about these issues. Placental endothelin 1 (*EDN1*) mRNA data for four different groups of participants is presented in bar graphs showing mean ± SE (Panel A), or mean ± SD (Panel B), and in a univariate scatterplot (Panel C). Panel A (mean ± SE) suggests that the second group has higher values than the remaining groups; however, Panel B (mean ± SD) reveals that there is considerable overlap between groups. Showing SE rather than SD magnifies the apparent visual differences between groups, and this is exacerbated by the fact that SE obscures any effect of unequal sample size. The scatterplot (Panel C) clearly shows that the sample sizes are small, group one has a much larger variance than the other groups, and there is an outlier in group three. These problems are not apparent in the bar graphs shown in Panels A and B.

doi:10.1371/journal.pbio.1002128.g003

Weissgerber TL, et al. (2015) Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm. PLOS Biology 13(4): e1002128.

Bar charts with error bars

What is wrong with {bar charts + error bars} ?

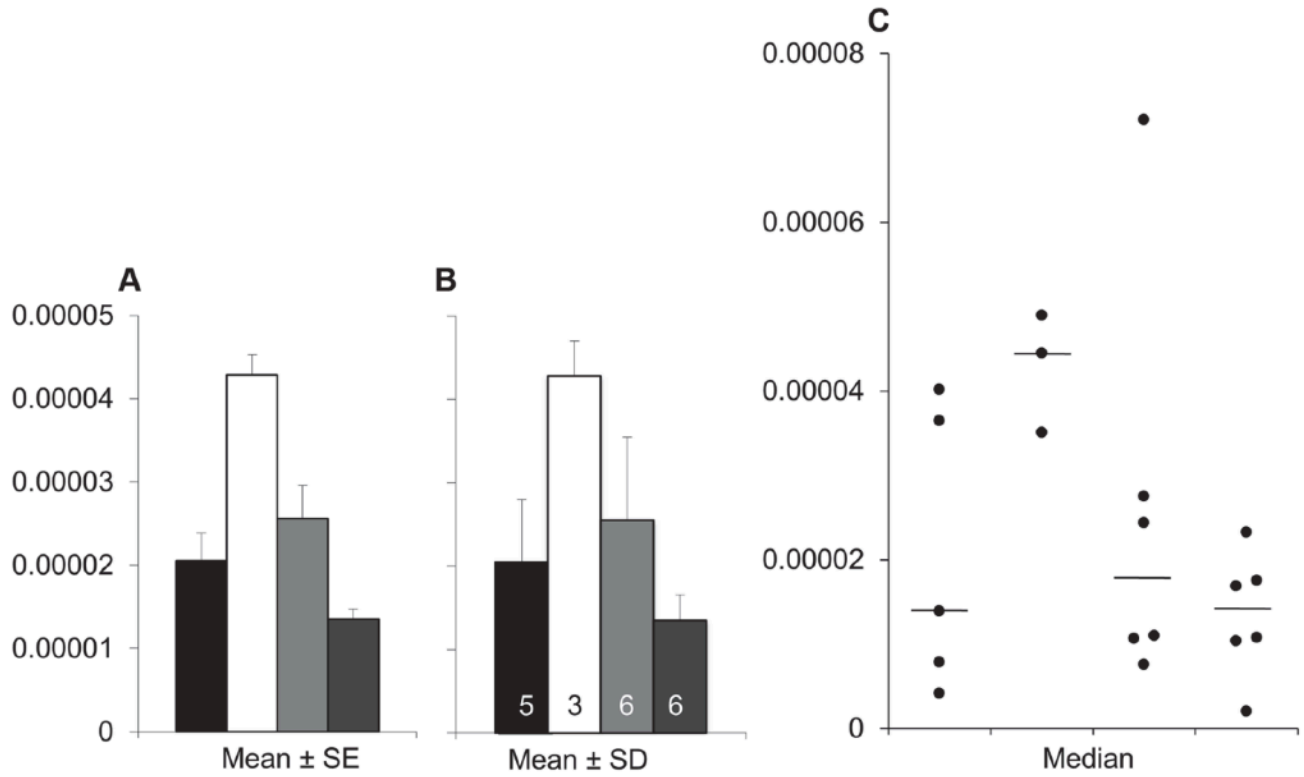


Fig 3. Bar graphs and scatterplots convey very different information. While scatterplots prompt the reader to critically evaluate the statistical tests and the authors' interpretation of the data, bar graphs discourage the reader from thinking about these issues. Placental endothelin 1 (*EDN1*) mRNA data for four different groups of participants is presented in bar graphs showing mean ± SE (Panel A), or mean ± SD (Panel B), and in a univariate scatterplot (Panel C). Panel A (mean ± SE) suggests that the second group has higher values than the remaining groups; however, Panel B (mean ± SD) reveals that there is considerable overlap between groups. Showing SE rather than SD magnifies the apparent visual differences between groups, and this is exacerbated by the fact that SE obscures any effect of unequal sample size. The scatterplot (Panel C) clearly shows that the sample sizes are small, group one has a much larger variance than the other groups, and there is an outlier in group three. These problems are not apparent in the bar graphs shown in Panels A and B.

doi:10.1371/journal.pbio.1002128.g003

Weissgerber TL, et al. (2015) Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm. PLOS Biology 13(4): e1002128.

Bar charts with error bars

What is wrong with {bar charts + error bars} ?

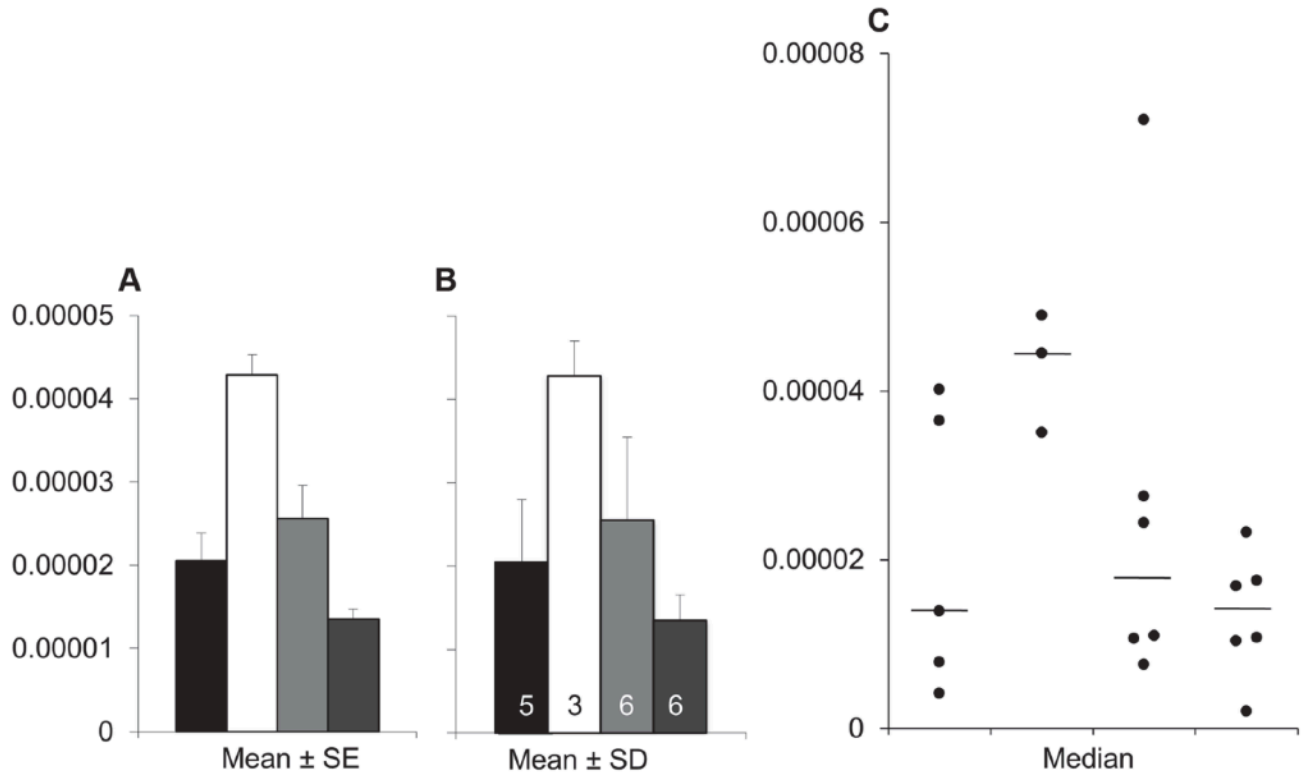


Fig 3. Bar graphs and scatterplots convey very different information. While scatterplots prompt the reader to critically evaluate the statistical tests and the authors' interpretation of the data, bar graphs discourage the reader from thinking about these issues. Placental endothelin 1 (*EDN1*) mRNA data for four different groups of participants is presented in bar graphs showing mean ± SE (Panel A), or mean ± SD (Panel B), and in a univariate scatterplot (Panel C). Panel A (mean ± SE) suggests that the second group has higher values than the remaining groups; however, Panel B (mean ± SD) reveals that there is considerable overlap between groups. Showing SE rather than SD magnifies the apparent visual differences between groups, and this is exacerbated by the fact that SE obscures any effect of unequal sample size. The scatterplot (Panel C) clearly shows that the sample sizes are small, group one has a much larger variance than the other groups, and there is an outlier in group three. These problems are not apparent in the bar graphs shown in Panels A and B.

doi:10.1371/journal.pbio.1002128.g003

Weissgerber TL, et al. (2015) Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm. PLOS Biology 13(4): e1002128.

Bar charts (with error bars) **not good for showing distributions**

Use bar charts **only to show class counts.**

~~Bar charts with error bars~~

What is wrong with {bar charts + error bars} ?

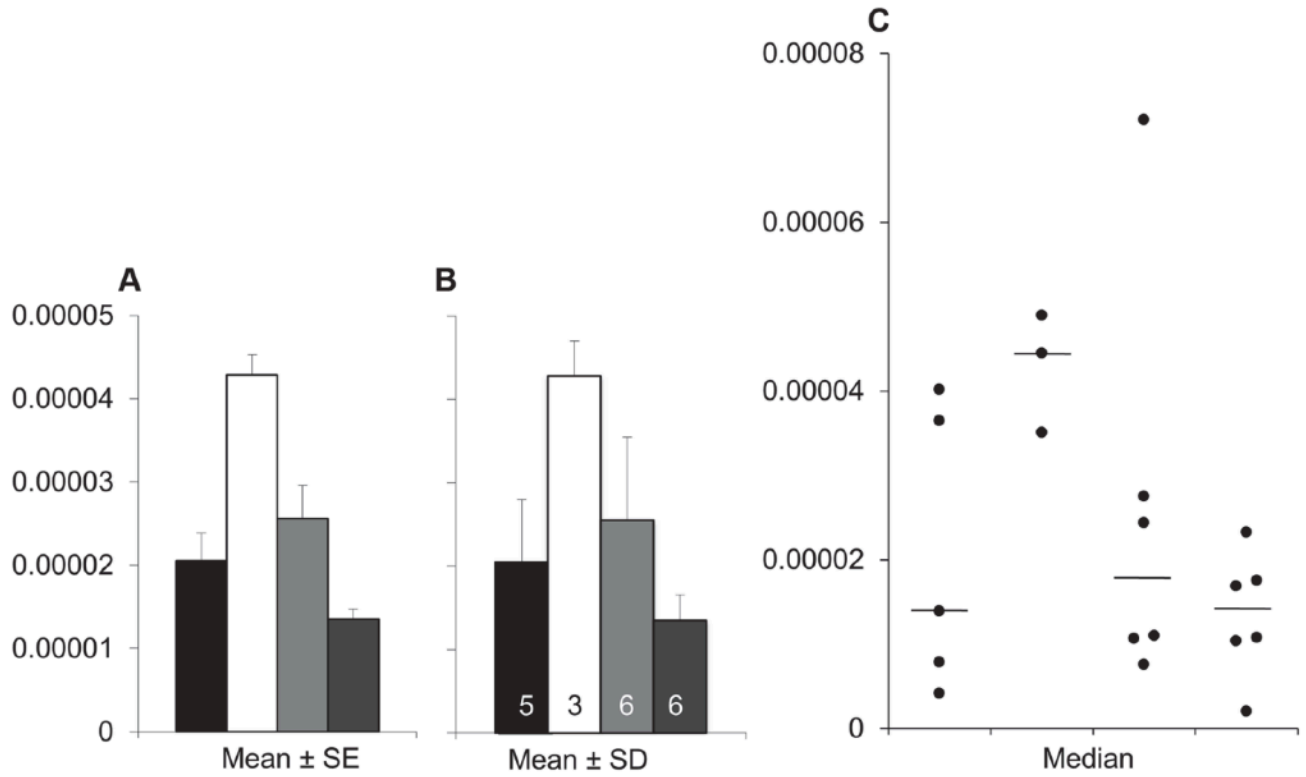


Fig 3. Bar graphs and scatterplots convey very different information. While scatterplots prompt the reader to critically evaluate the statistical tests and the authors' interpretation of the data, bar graphs discourage the reader from thinking about these issues. Placental endothelin 1 (*EDN1*) mRNA data for four different groups of participants is presented in bar graphs showing mean ± SE (Panel A), or mean ± SD (Panel B), and in a univariate scatterplot (Panel C). Panel A (mean ± SE) suggests that the second group has higher values than the remaining groups; however, Panel B (mean ± SD) reveals that there is considerable overlap between groups. Showing SE rather than SD magnifies the apparent visual differences between groups, and this is exacerbated by the fact that SE obscures any effect of unequal sample size. The scatterplot (Panel C) clearly shows that the sample sizes are small, group one has a much larger variance than the other groups, and there is an outlier in group three. These problems are not apparent in the bar graphs shown in Panels A and B.

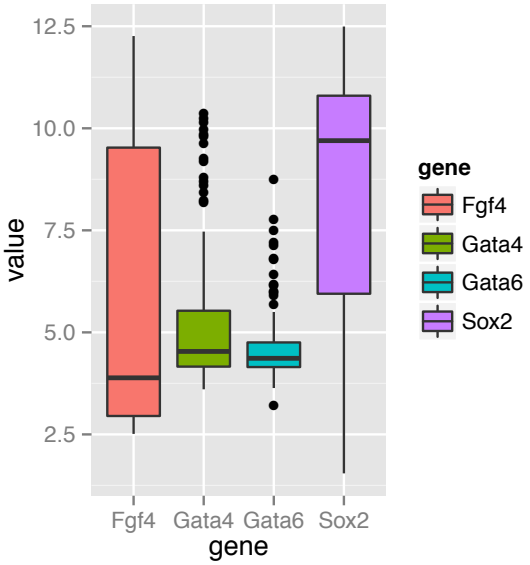
doi:10.1371/journal.pbio.1002128.g003

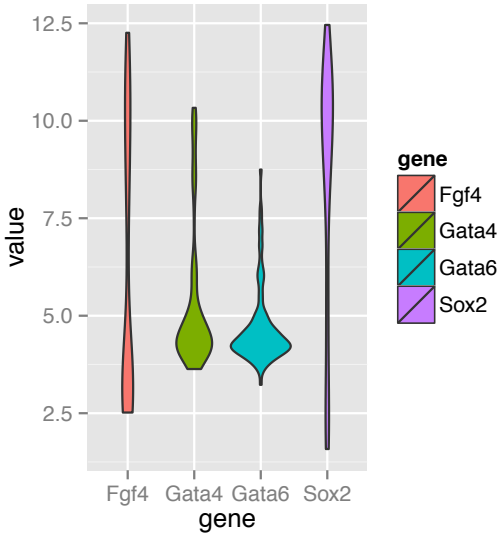
Weissgerber TL, et al. (2015) Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm. PLOS Biology 13(4): e1002128.

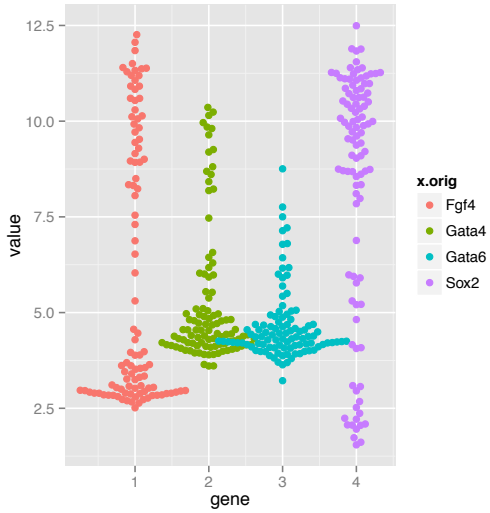
Bar charts (with error bars) **not good for showing distributions**

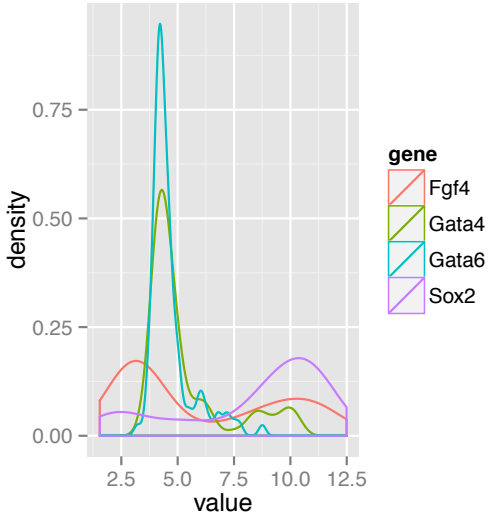
Use bar charts **only to show class counts.**

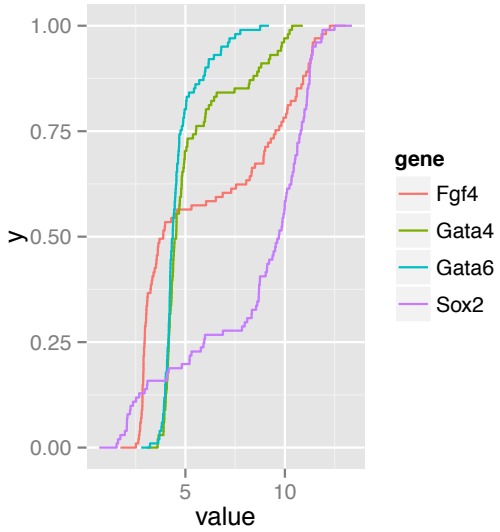
Summary: Visualizing distributions in 1D











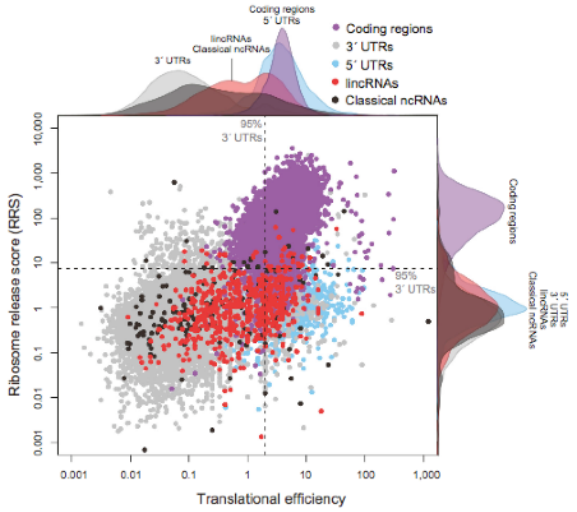
2D data plots

2D data plots

Scatterplots (x,y)-point plots

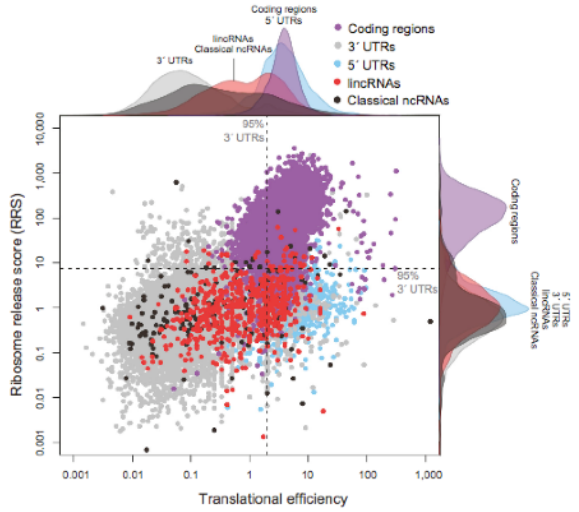
2D data plots

Scatterplots (x,y)-point plots

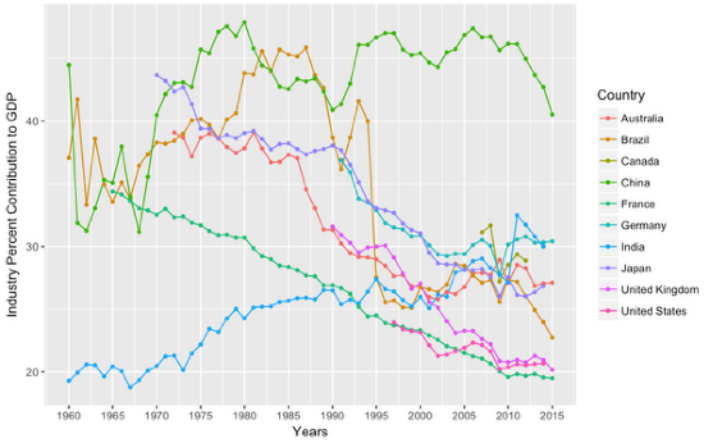


2D data plots

Scatterplots (x,y)-point plots

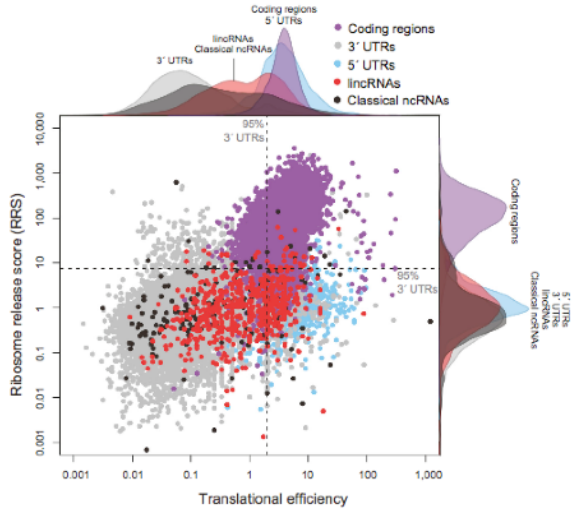


Line plots (x,y)-line plots

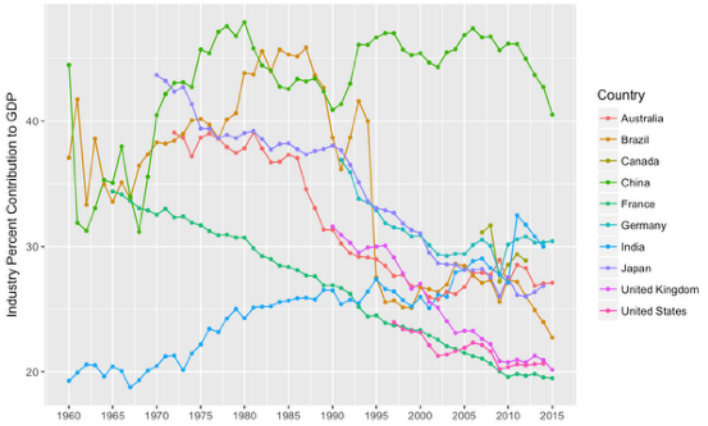


2D data plots

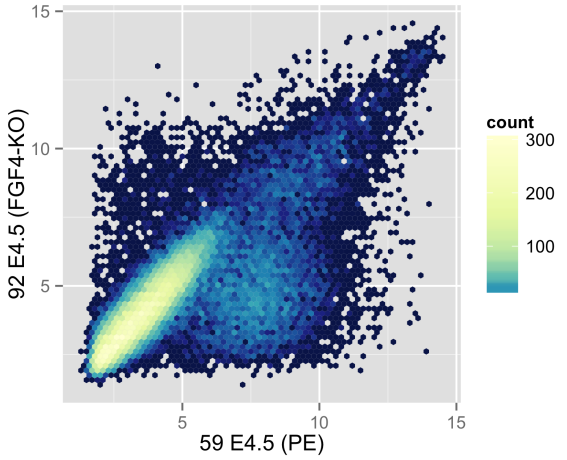
Scatterplots (x,y)-point plots



Line plots (x,y)-line plots

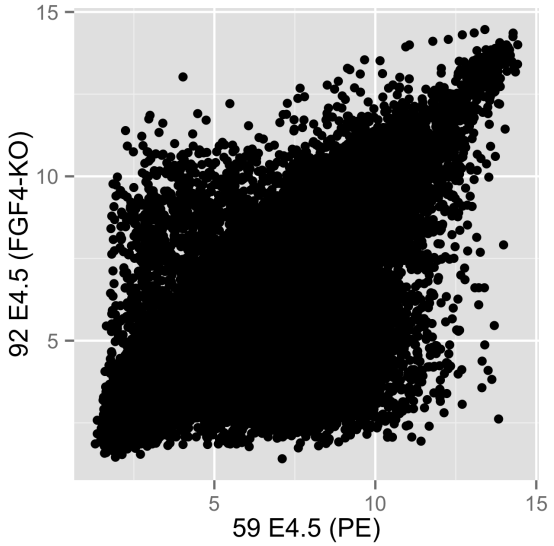


2D density requires the choice of bandwidth; obscures the sample size (i.e. the uncertainty of the estimate)



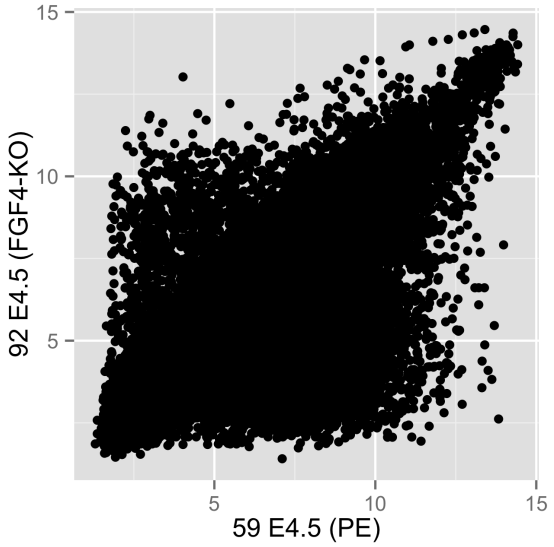
Showing distributions in 2D

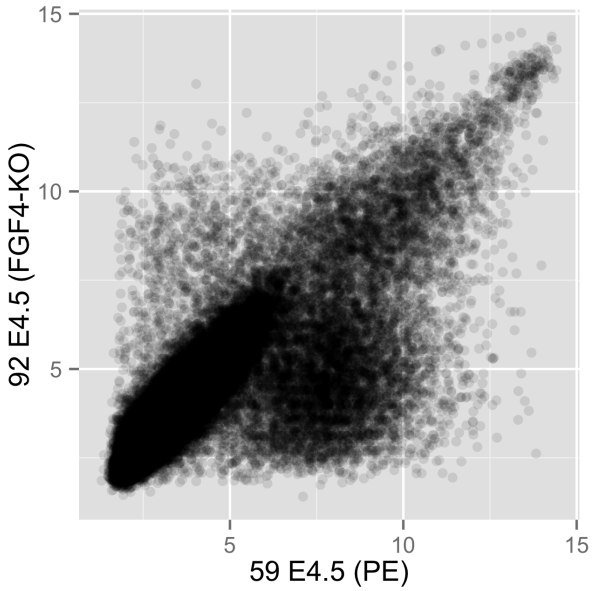
```
scp <- ggplot(dfx, aes( x = ‘59 E4.5 (PE)‘ ,
                        y = ‘92 E4.5 (FGF4-KO)‘))
scp + geom_point()
```



Showing distributions in 2D

```
scp <- ggplot(dfx, aes( x = ‘59 E4.5 (PE)‘ ,
                        y = ‘92 E4.5 (FGF4-KO)‘))
scp + geom_point()
```

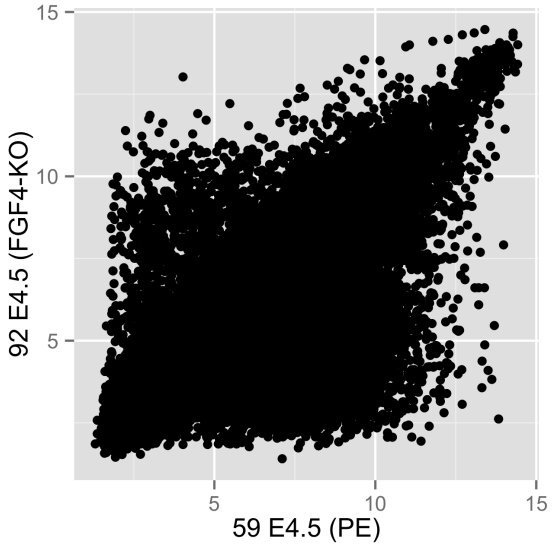


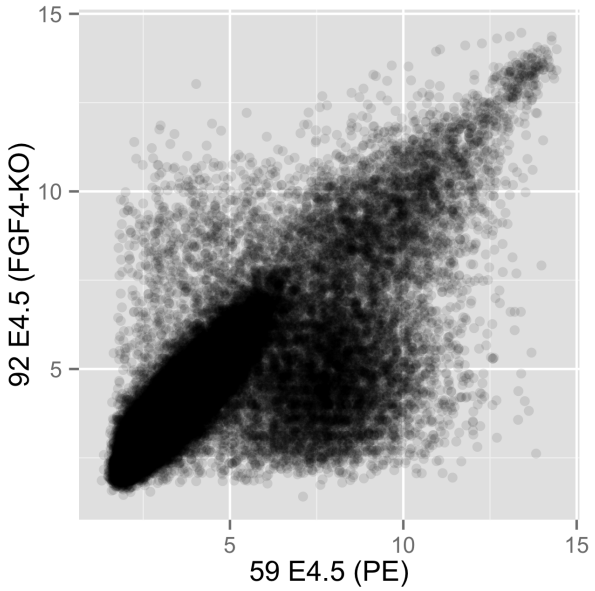



```
scp  + geom_point(alpha = 0.1)
```

Showing distributions in 2D

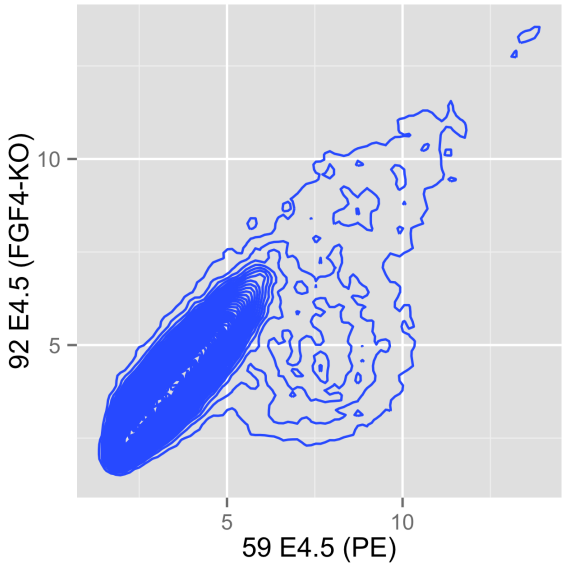
```
scp <- ggplot(dfx, aes( x = '59 E4.5 (PE)' ,
                        y = '92 E4.5 (FGF4-KO)'))
scp + geom_point()
```



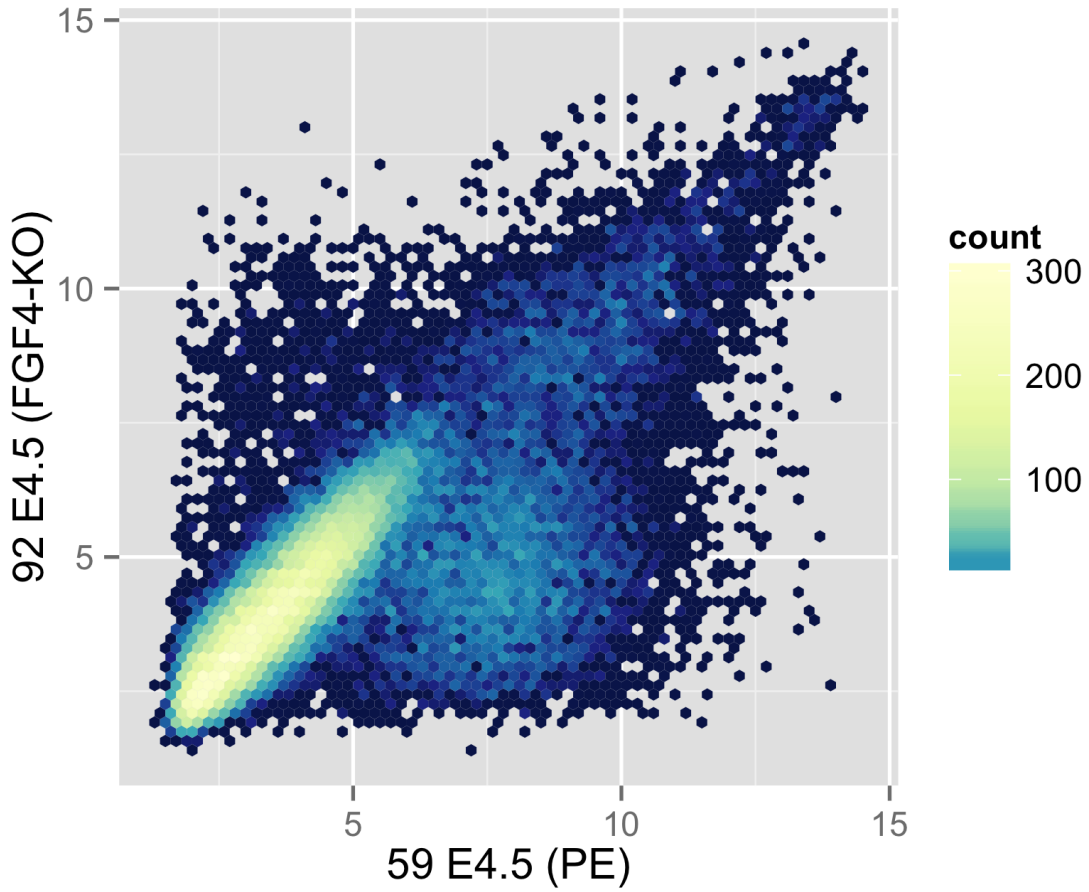



```
scp  + geom_point(alpha = 0.1)
```

```
scp + geom_density2d(h = 0.5, bins = 60)
```



binhex is a good, easy to read, option to show 2D density




```
scp + stat_binhex(binwidth = c(0.2, 0.2)) + colourscale +
  coord_fixed()
```

How to show more than 2D?

How to show more than 2D?

What tricks do you use for >2D?

3-5D: **aesthetics** allow to show more than 2D

geom_point's

aesthetics

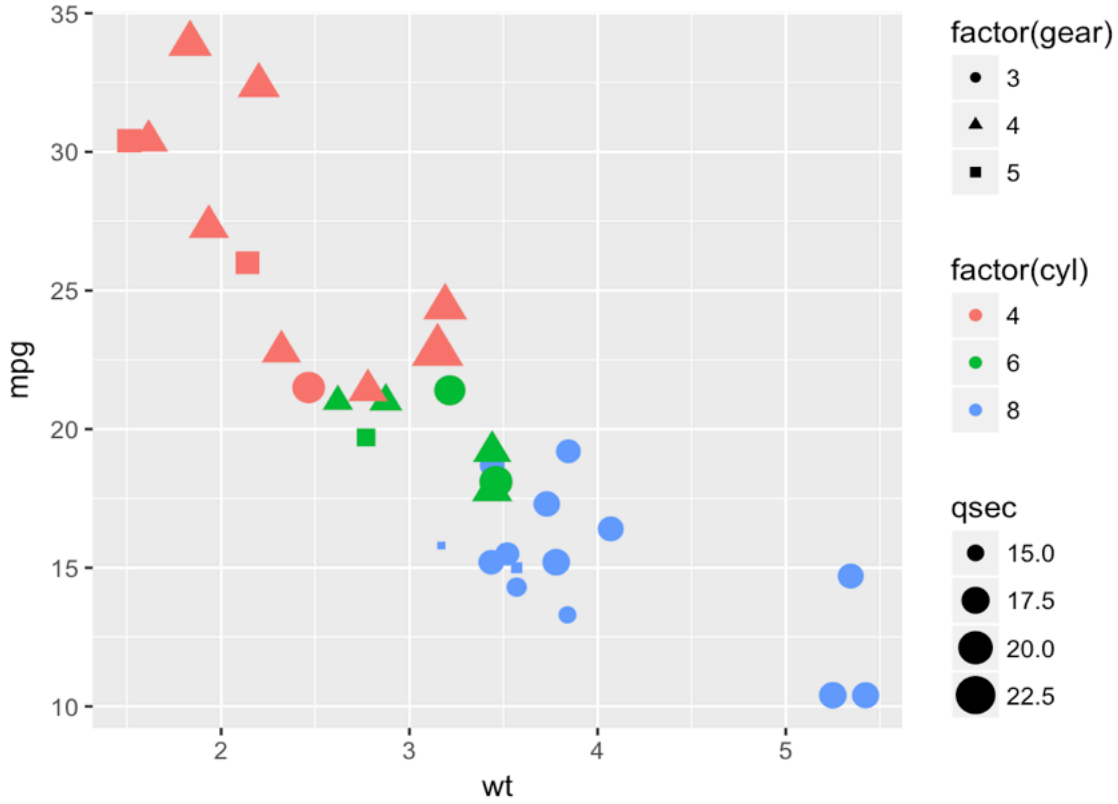
(apart from x and y):

- fill / color
- shape
- size
- alpha

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
ggplot(data = mtcars) +
  geom_point(
    aes(x = wt, y = mpg,
        shape = factor(gear),
        color = factor(cyl),
        size = qsec))
```



3-5D: **aesthetics** allow to show more than 2D

geom_point's

aesthetics

(apart from x and y):

- fill / color
- shape
- size
- alpha

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```


```
ggplot(data = mtcars) +
  geom_point(
    aes(x = wt, y = mpg,
        shape = factor(gear),
        color = factor(cyl),
        size = qsec))
```

3-5D: **aesthetics** allow to show more than 2D

geom_point's

aesthetics

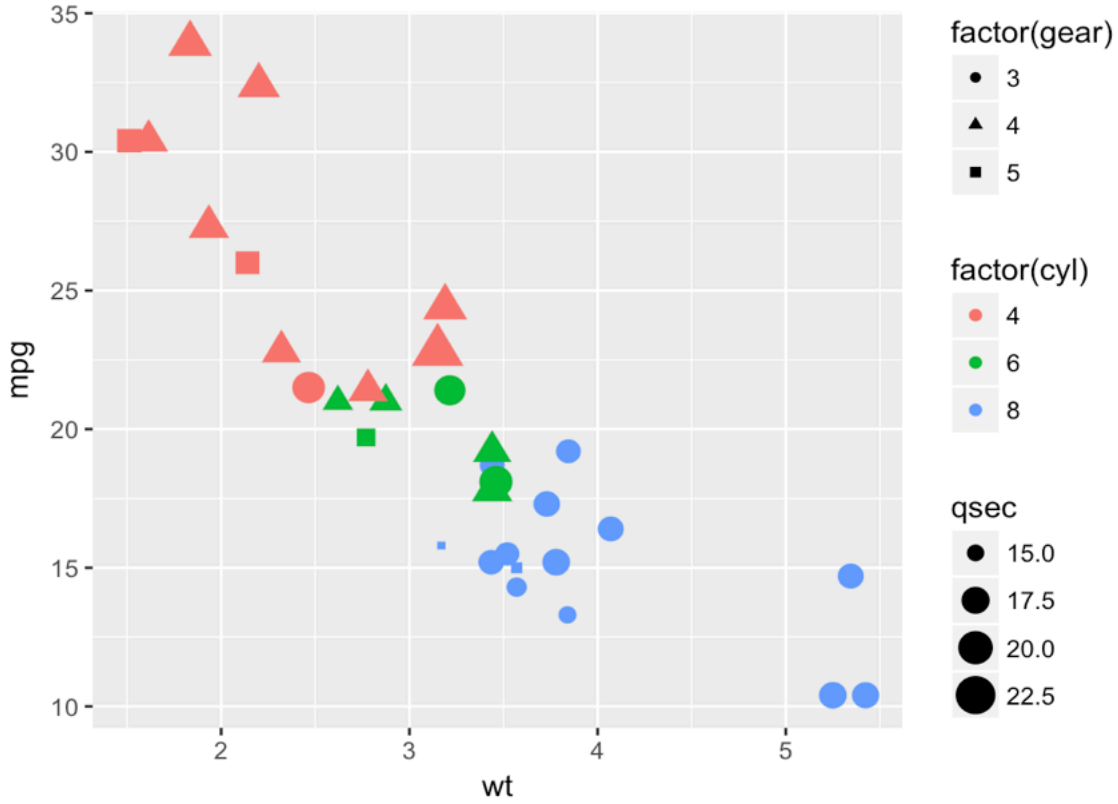
(apart from x and y):

- fill / color
- shape
- size
- alpha

```
head(mtcars)
```

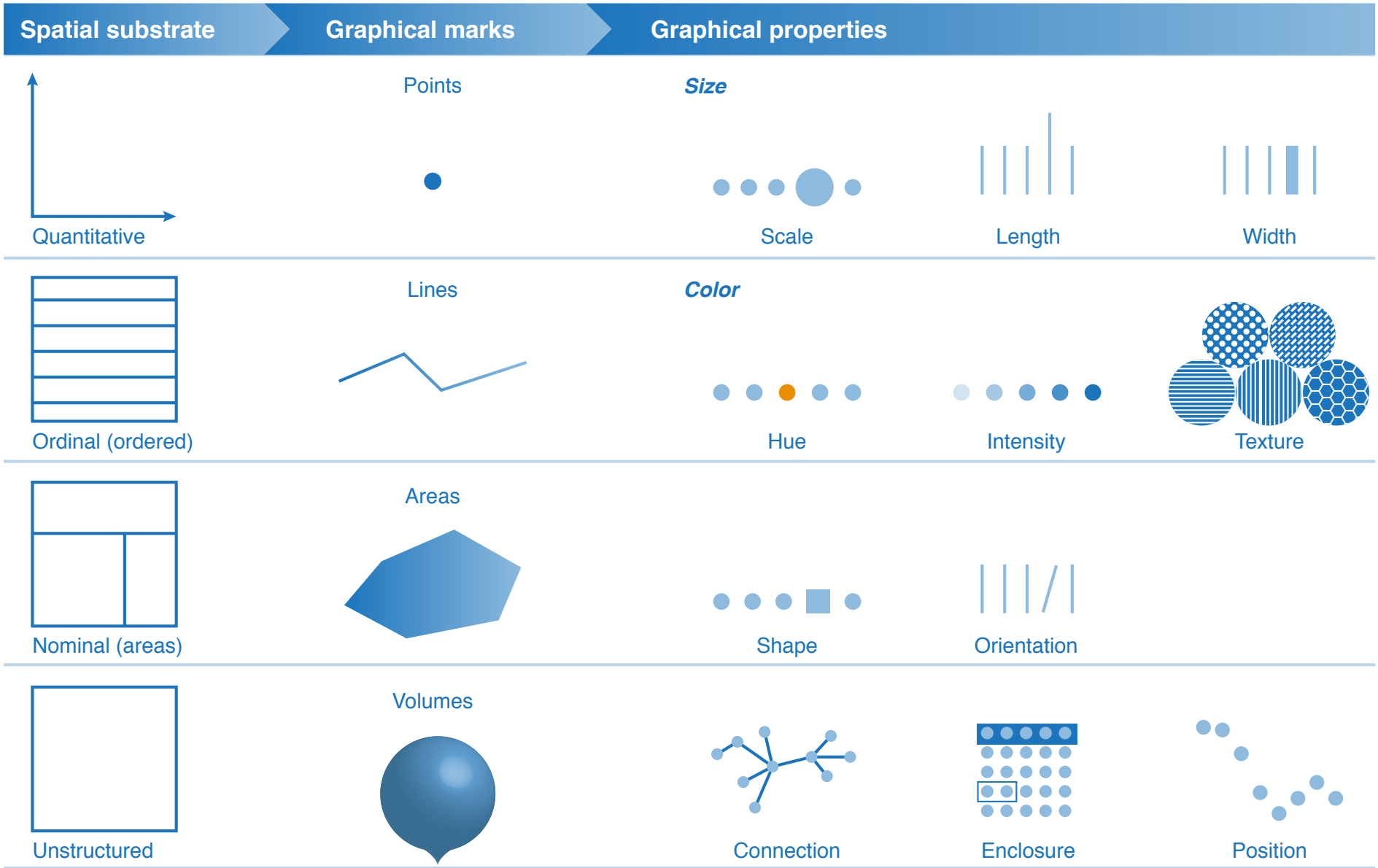
```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
ggplot(data = mtcars) +
  geom_point(
    aes(x = wt, y = mpg,
        shape = factor(gear),
        color = factor(cyl),
        size = qsec))
```



A diversity of **graphical properties (aesthetics)** are available to show dimensions

A diversity of **graphical properties (aesthetics)** are available to show dimensions

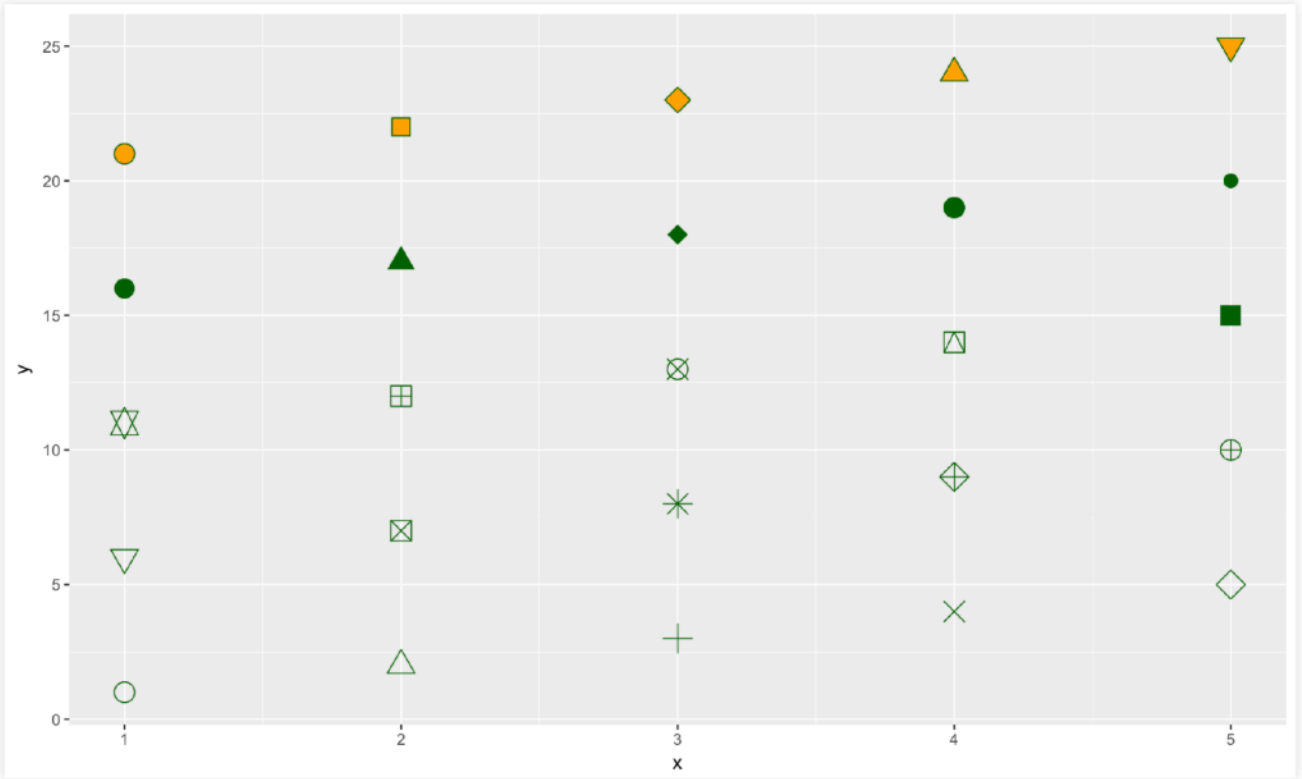


Marker shapes and colors in R

geom_point's

aesthetics (beyond x and y):

- **fill / color**
- **shape**
- size
- alpha

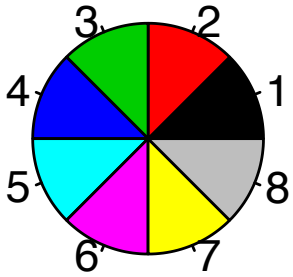



```
ggplot(data.frame(x = 1:5 , y = 1:25, z = 1:25), aes(x = x, y = y)) +
  geom_point(aes(shape = z), size = 5, colour = "darkgreen", fill = "orange") +
  scale_shape_identity()
```

Color Usage

Default color scheme in base R plot:

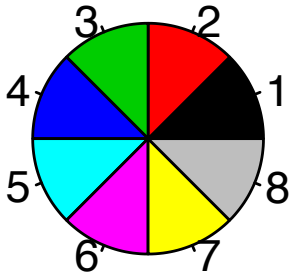
```
pie(rep(1, 8), col=1:8)
```



Color Usage

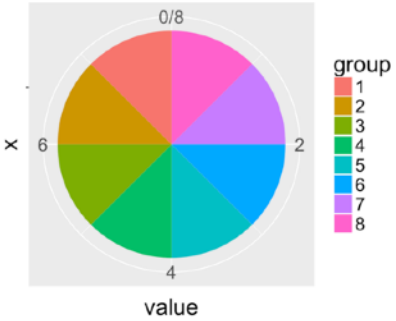
Default color scheme in base R plot:

```
pie(rep(1, 8), col=1:8)
```



Default color scheme in ggplot:

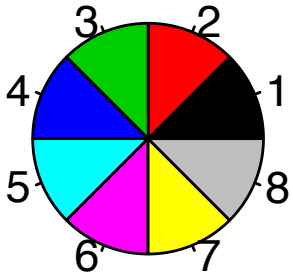
```
ggplot(data.frame(group = factor(seq_len(8)), value = rep(1, 8)),
       aes(x="", y=value, fill=group)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  theme(text = element_text(size = 20))
```



Color Usage

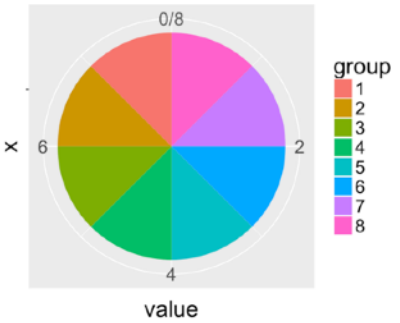
Default color scheme in base R plot:

```
pie(rep(1, 8), col=1:8)
```



Default color scheme in ggplot:

```
ggplot(data.frame(group = factor(seq_len(8)), value = rep(1, 8)),
       aes(x="", y=value, fill=group)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  theme(text = element_text(size = 20))
```

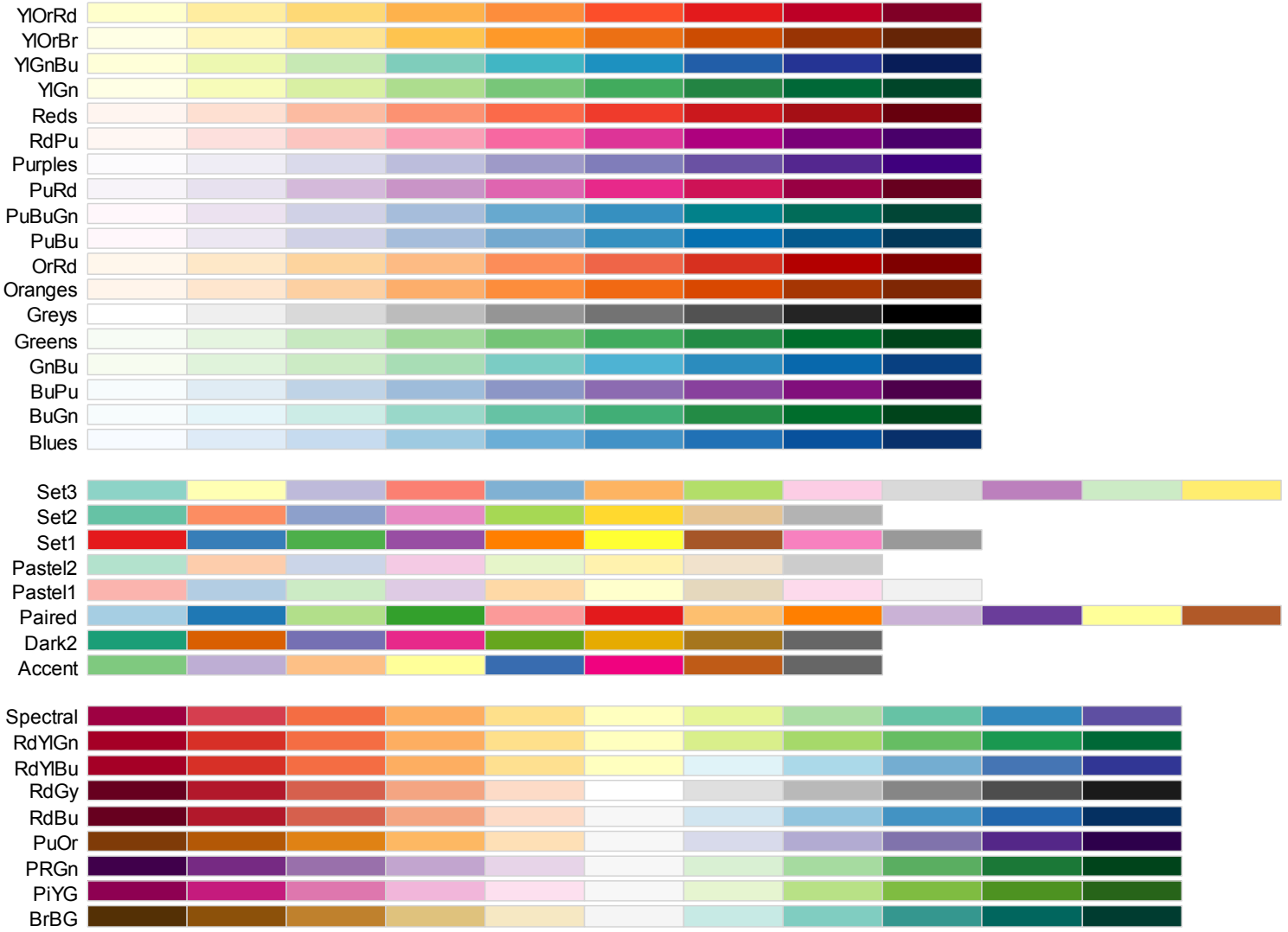


When choosing a coloring scheme, consider these:

- Different requirements for line & area colors
- Many people are **red-green color-blind**
- Lighter colors tend to make areas look larger than darker colors
  → **use colors of equal luminance for filled areas**.

RColorBrewer

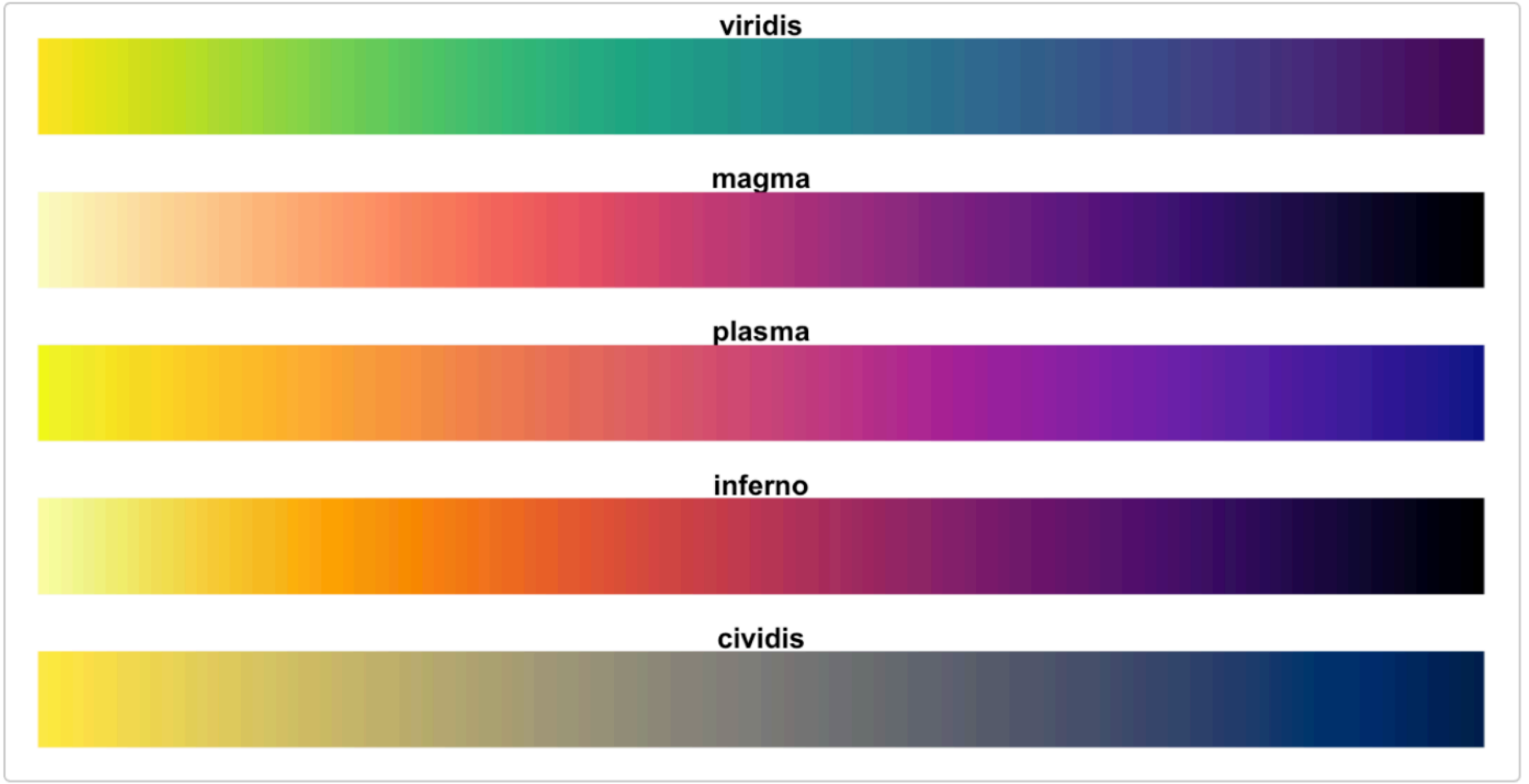
```
display.brewer.all()
```



Viridis Palettes

```
install.packages("viridis")
library(viridis)
```

Simply add: `scale_color_viridis()` `scale_fill_viridis()`. to your plot

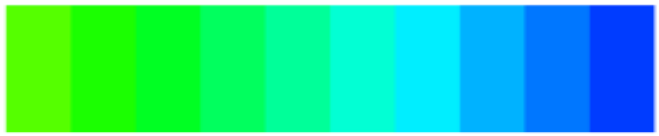


Viridis Palettes

Color scales are designed to be:

- **Colorful and Pretty**, spanning as wide a palette as possible so as to make differences easy to see,
- **Perceptually uniform**, the perceived difference between two colors is proportional to the Euclidian distance within the color space
- **Robust to colorblindness**, looks good in grey scale and to people with common forms of colorblindness

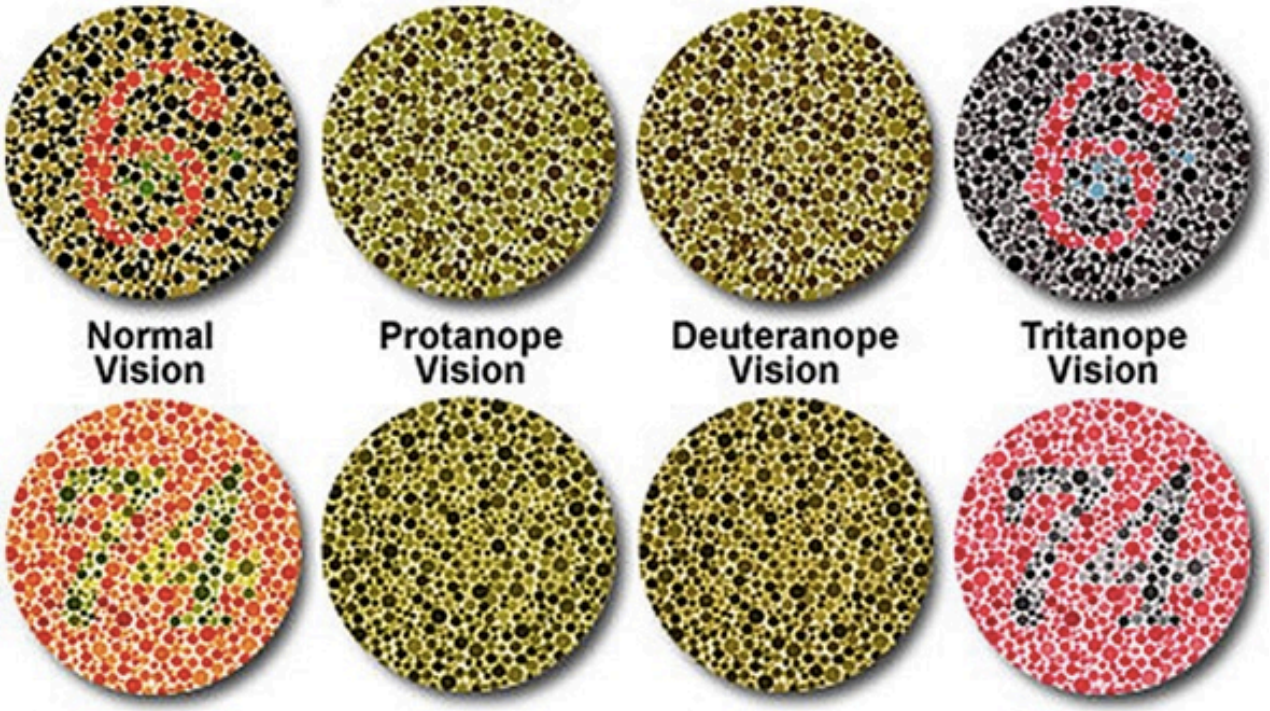
You can hear more about the science behind creating these color scales, on Walt and Smith's talk at SciPy 2015.

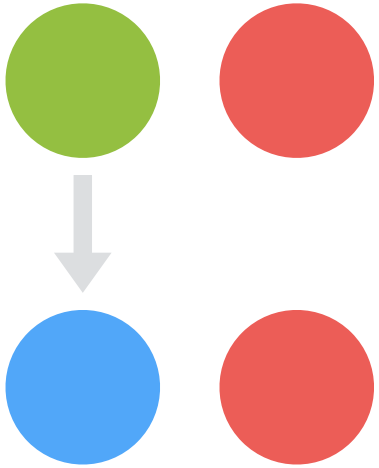


not perceptually uniform

perceptually uniform

Be kind to colorblind people





Simple solution: replace greens by blues.
Blues also display better on most monitors than greens.

Color models

Color models

How are colors defined?

Color models

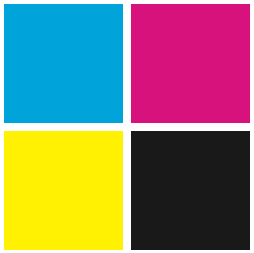
How are colors defined?

RGB

Color models

How are colors defined?

RGB

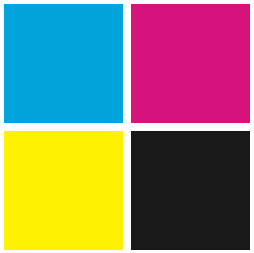


CMYK

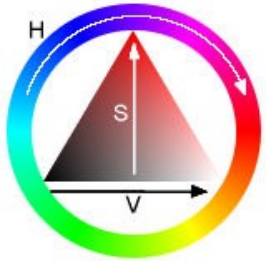
Color models

How are colors defined?

RGB



CMYK



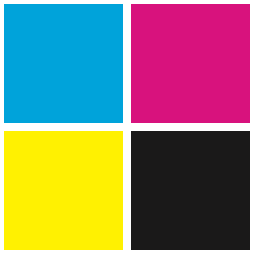
HSV
HSB

Color models

How are colors defined?

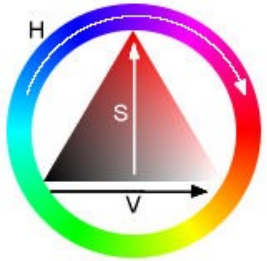
RGB

light



CMYK

ink



HSV
HSB

human perception

Faceting is useful to show more dimensions without overcrowding the graph

Faceting is useful to show more dimensions without overcrowding the graph

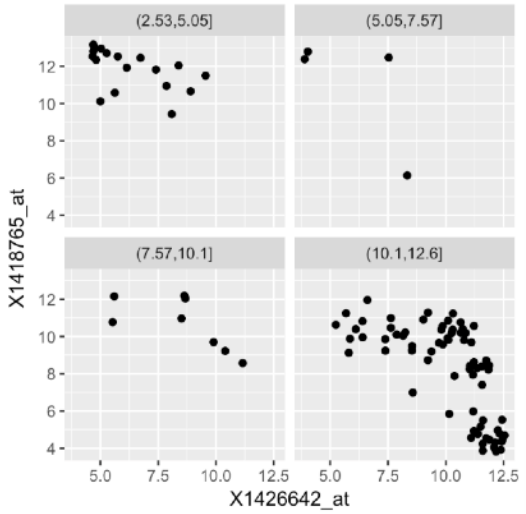


Figure 3.33: **Faceting**: the same data as in Figure 3.9, split by the continuous variable `X1450989_at` and arranged by `facet_wrap`.

Trellis — chart that uses multiple instances of the same chart

`facet_wrap`

```
ggplot(mutate(dftx, Tdgf1 = cut(X1450989_at, breaks = 4)),
   aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_wrap( ~ Tdgf1, ncol = 2 )
```

Faceting is useful to show more dimensions without overcrowding the graph

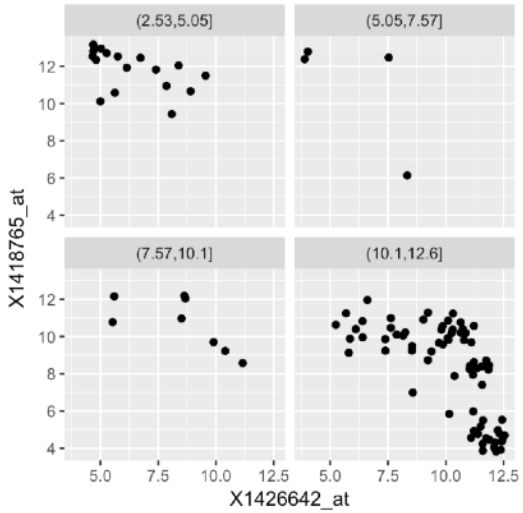


Figure 3.33: **Faceting**: the same data as in Figure 3.9, split by the continuous variable `X1450989_at` and arranged by `facet_wrap`.

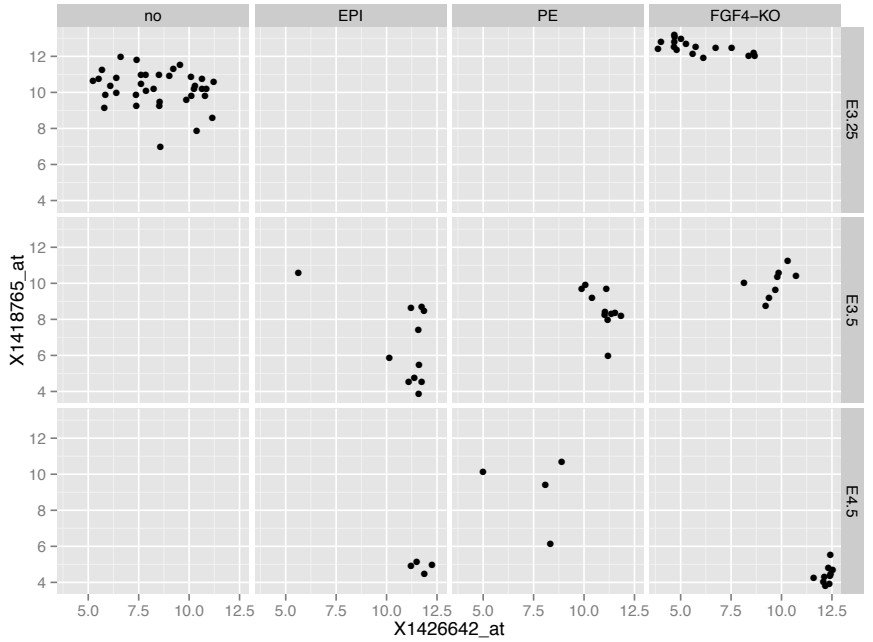
Trellis — chart that uses multiple instances of the same chart

`facet_wrap`

```
ggplot(mutate(dftx, Tdgf1 = cut(X1450989_at, breaks = 4)),
   aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_wrap( ~ Tdgf1, ncol = 2 )
```

`facet_grid`

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```



Faceting is useful to show more dimensions without overcrowding the graph

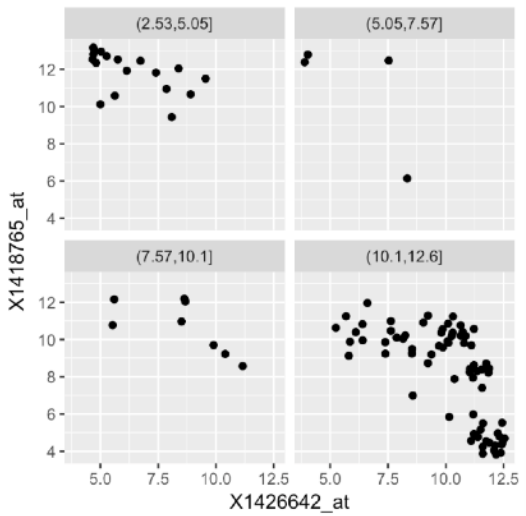


Figure 3.33: **Faceting**: the same data as in Figure 3.9, split by the continuous variable `X1450989_at` and arranged by `facet_wrap`.

Trellis — chart that uses multiple instances of the same chart

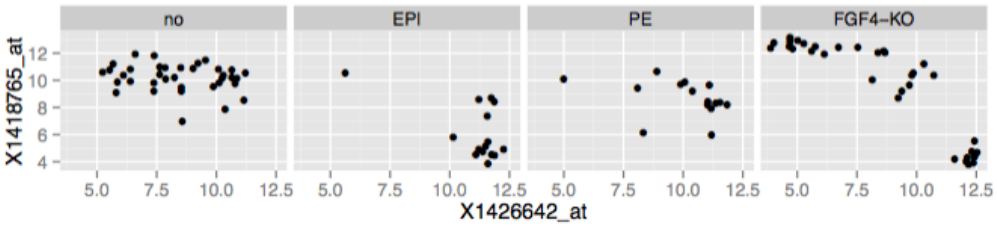
`facet_wrap`

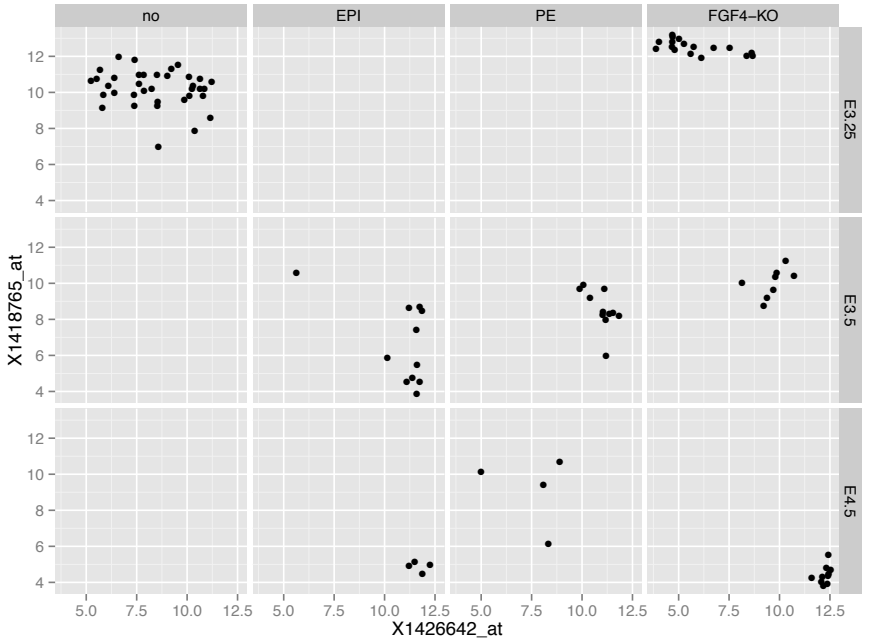
```
ggplot(mutate(dftx, Tdgf1 = cut(X1450989_at, breaks = 4)),
   aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_wrap( ~ Tdgf1, ncol = 2 )
```

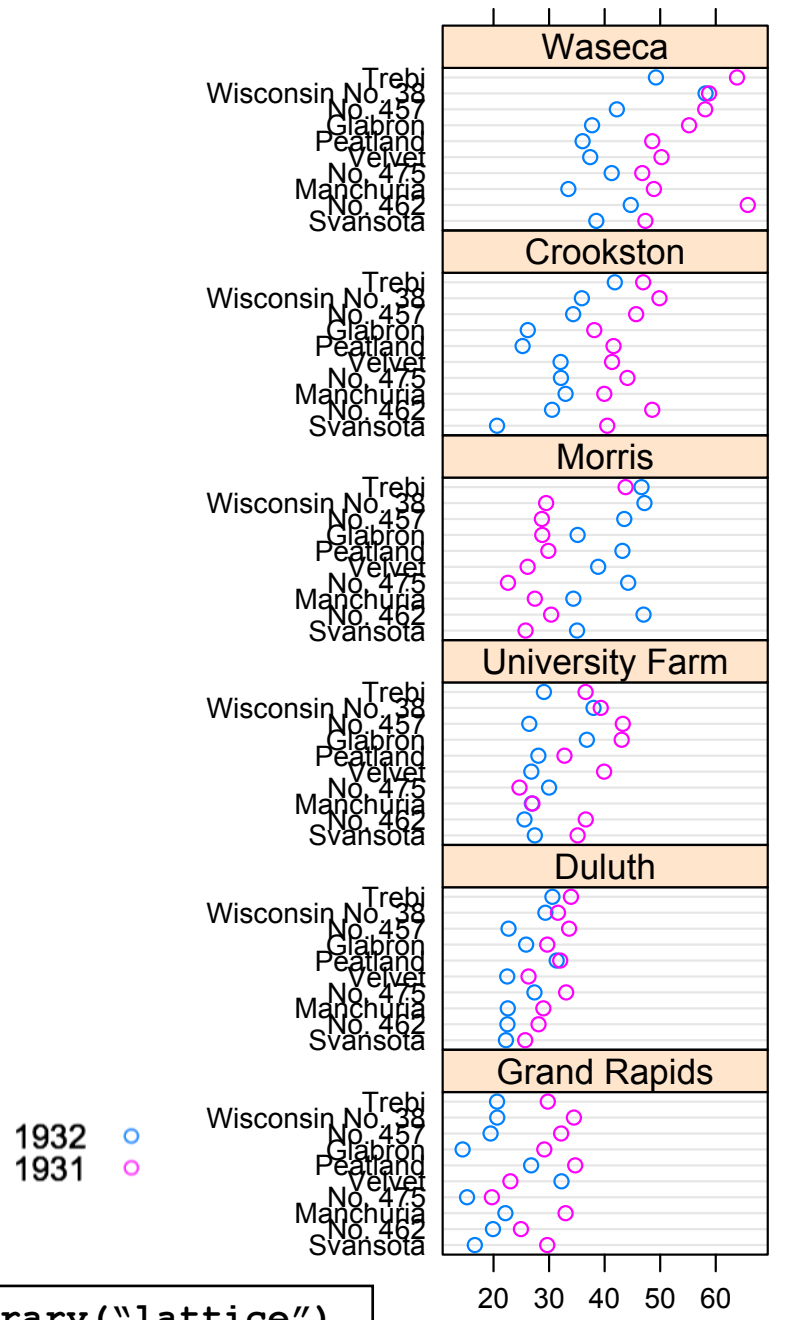
`facet_grid`

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

```
ggplot(dftx, aes( x = X1426642_at, y = X1418765_at)) +
  geom_point() + facet_grid( . ~ lineage )
```

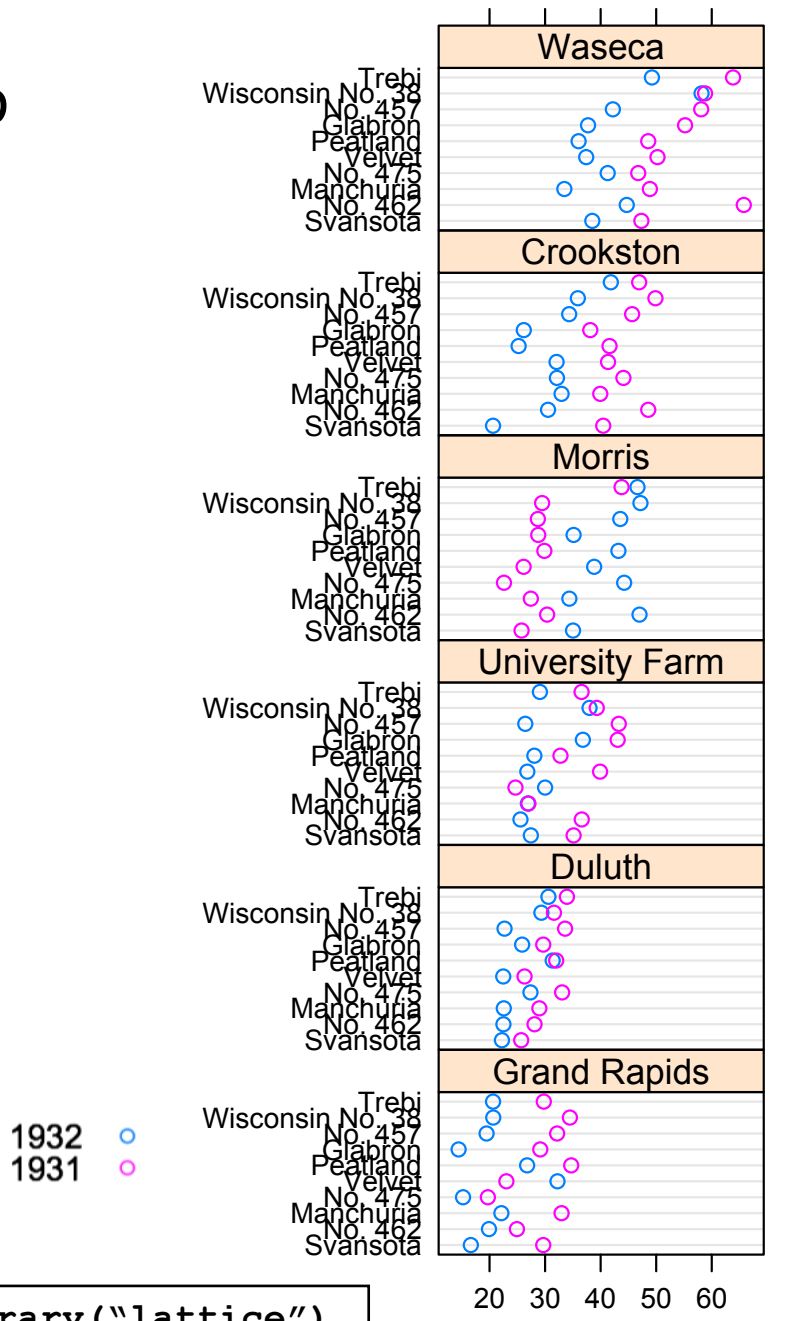







```
library("lattice")
example("barley")
```

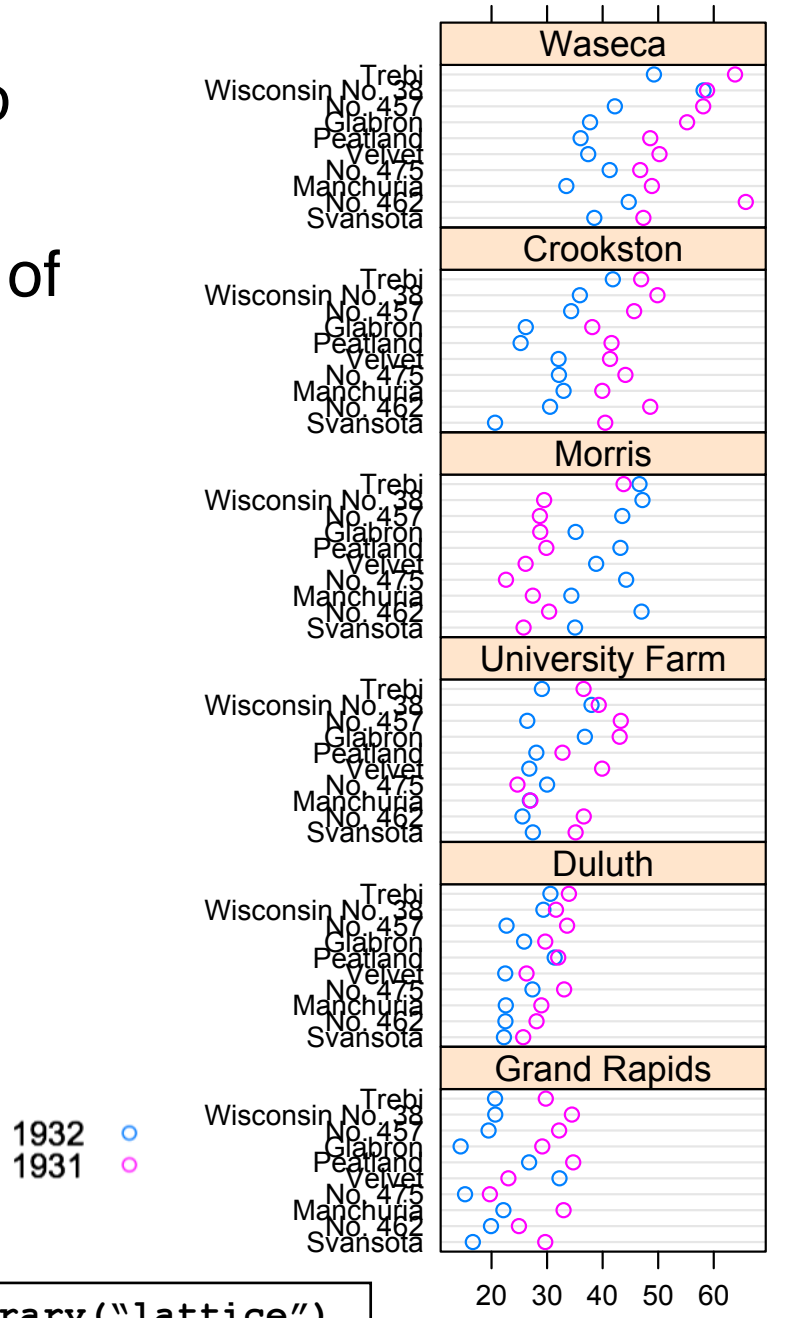
Data from an agricultural field trial to study the crop barley.




```
library("lattice")
example("barley")
```

Data from an agricultural field trial to study the crop barley.

At 6 sites in Minnesota, 10 varieties of barley were grown in each of two years.

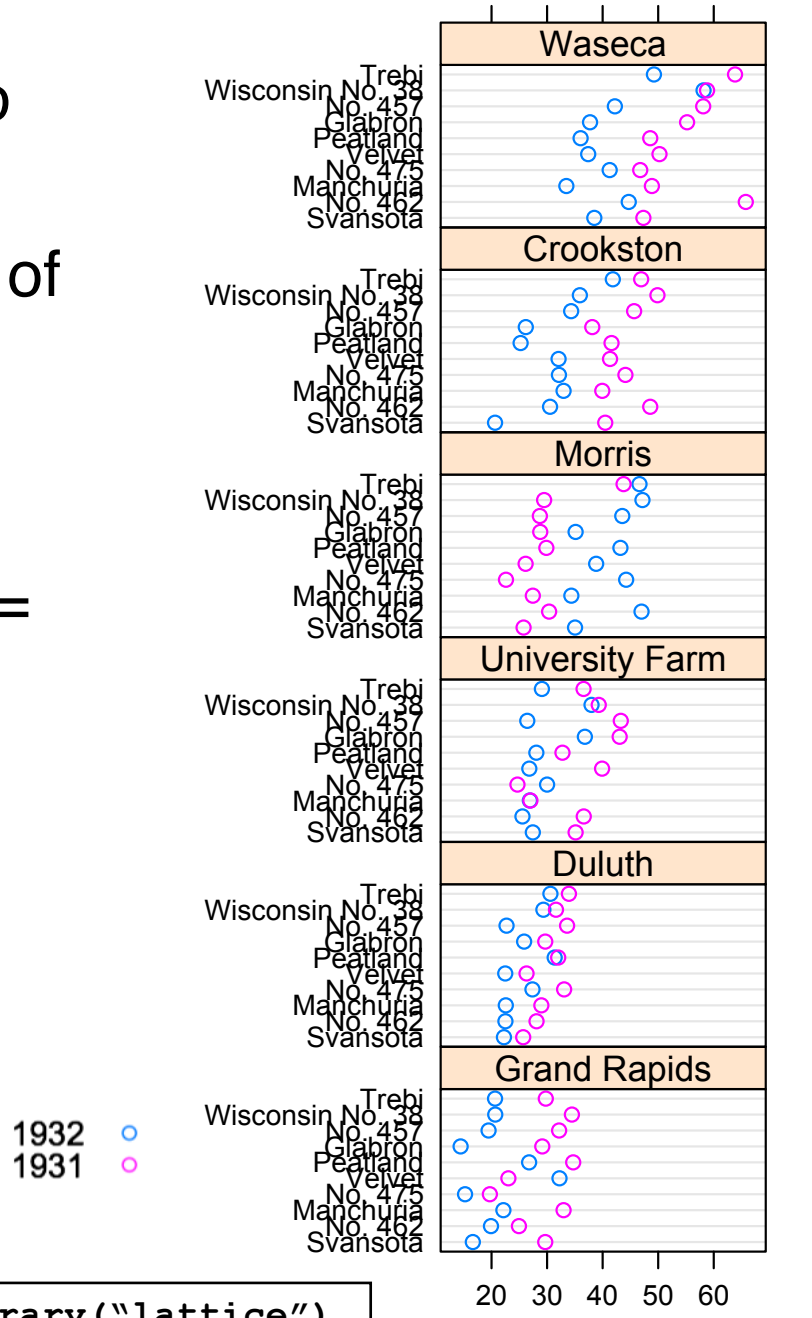



```
library("lattice")
example("barley")
```

Data from an agricultural field trial to study the crop barley.

At 6 sites in Minnesota, 10 varieties of barley were grown in each of two years.

Data: yield, for all combinations of site, variety, and year (6 x 10 x 2 = 120 observations)

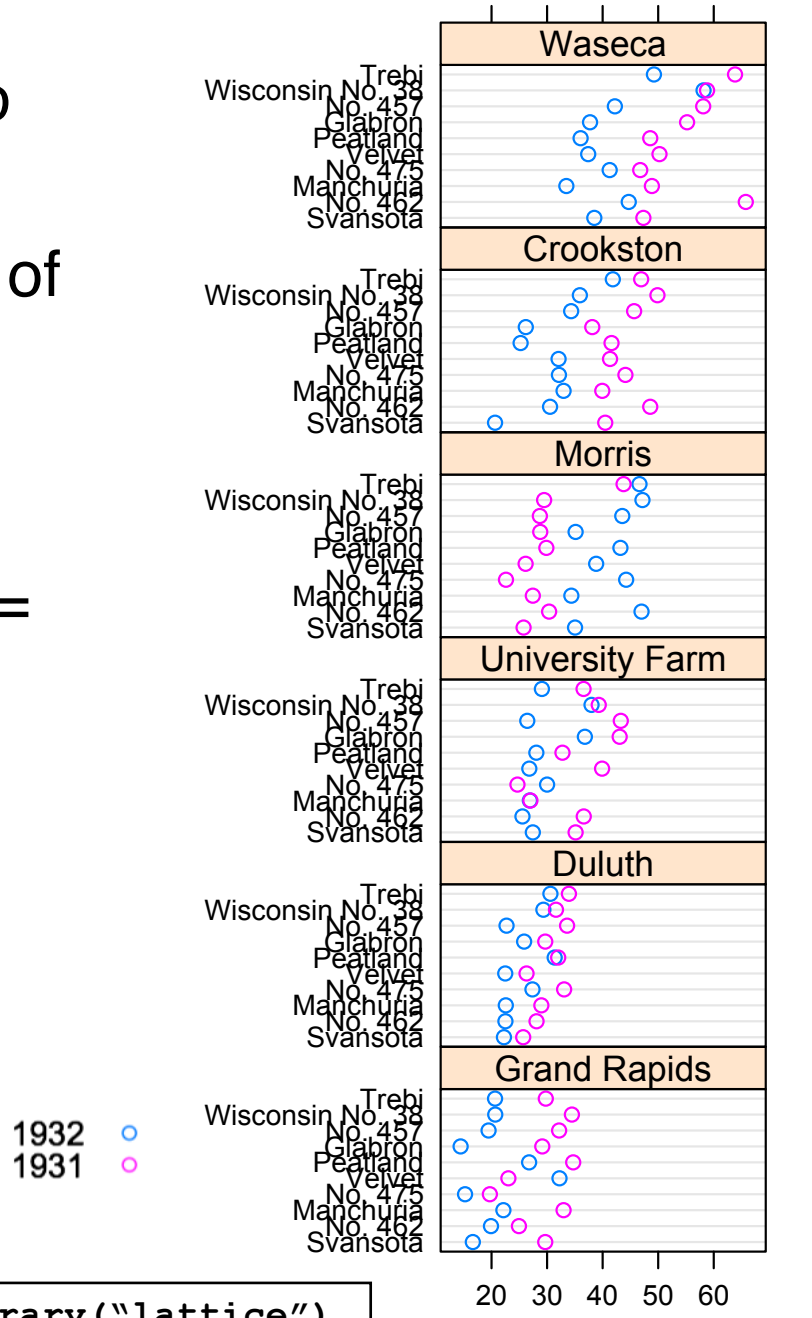



```
library("lattice")
example("barley")
```

Data from an agricultural field trial to study the crop barley.

At 6 sites in Minnesota, 10 varieties of barley were grown in each of two years.

Data: yield, for all combinations of site, variety, and year (6 x 10 x 2 = 120 observations)



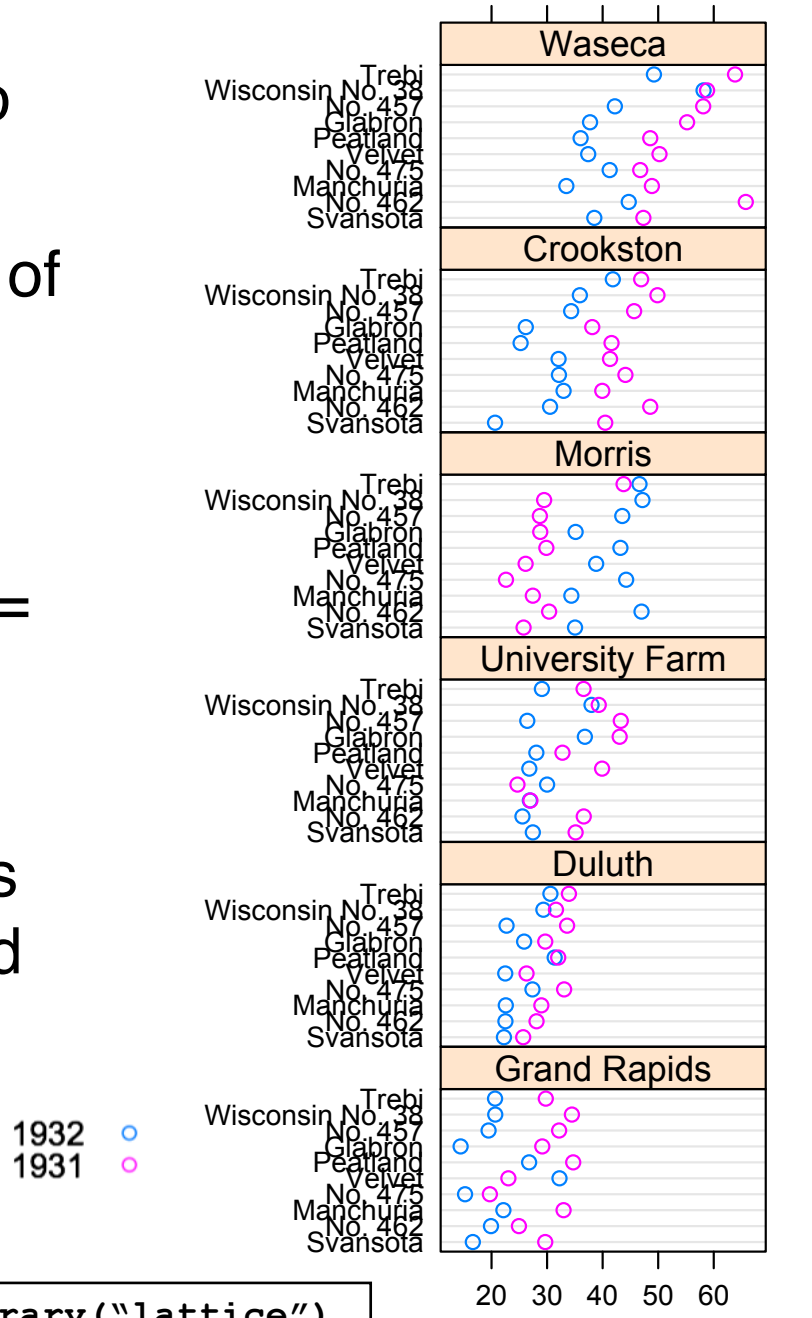

```
library("lattice")
example("barley")
```

Data from an agricultural field trial to study the crop barley.

At 6 sites in Minnesota, 10 varieties of barley were grown in each of two years.

Data: yield, for all combinations of site, variety, and year (6 x 10 x 2 = 120 observations)

Note the data for **Morris** - reanalysis in the 1990s using Trellis revealed that the years had been flipped!



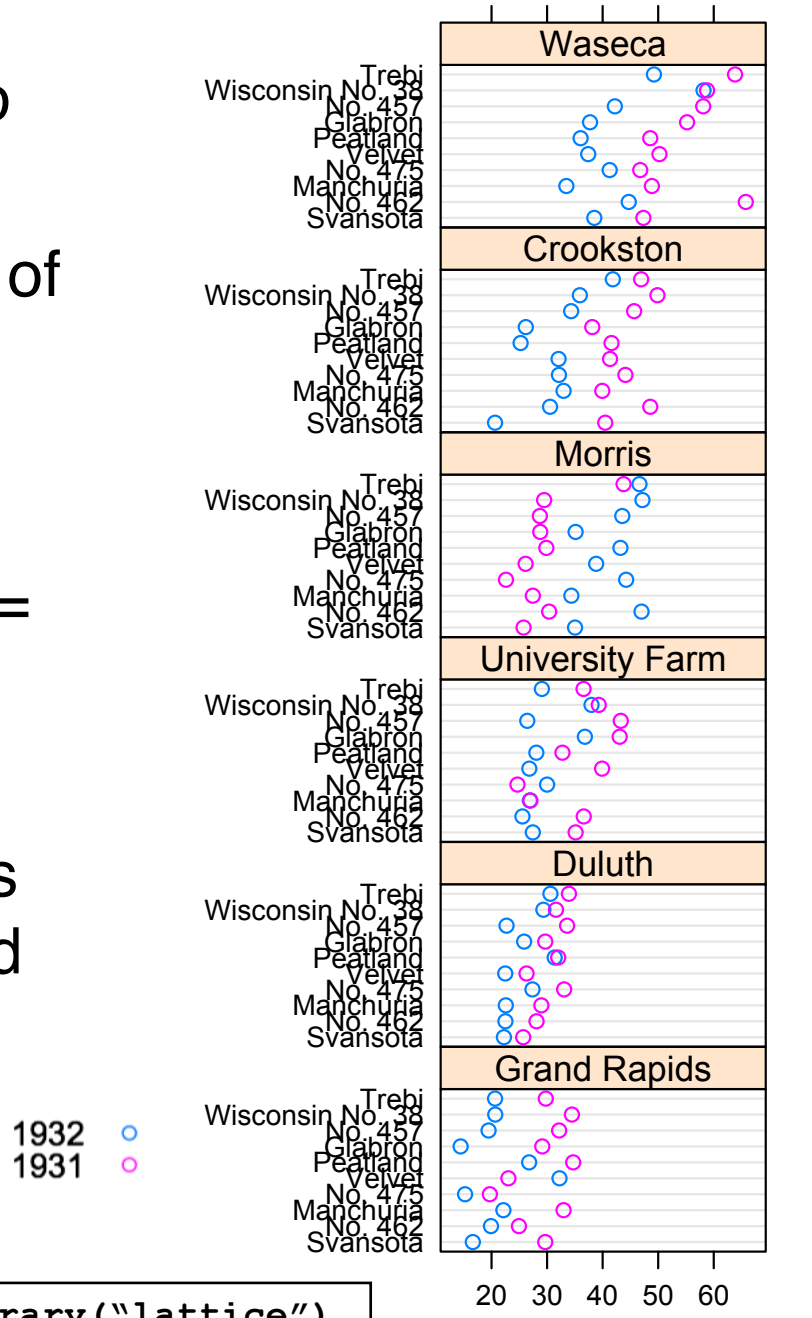

```
library("lattice")
example("barley")
```

Data from an agricultural field trial to study the crop barley.

At 6 sites in Minnesota, 10 varieties of barley were grown in each of two years.

Data: yield, for all combinations of site, variety, and year (6 x 10 x 2 = 120 observations)

Note the data for **Morris** - reanalysis in the 1990s using Trellis revealed that the years had been flipped!



How could you quickly check for potential batch effects?

```
library("lattice")
example("barley")
```

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Data.frame in R can be in:

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Data.frame in R can be in:

wide format

```
##         X1420085_at X1418863_at X1425463_at X1416967_at
## 1 E3.25    3.027715    4.843137    5.500618    1.731217
## 2 E3.25    9.293016    5.530016    6.160900    9.697038
## 3 E3.25    2.940142    4.418059    4.584961    4.161240
## 4 E3.25    9.715243    5.982314    4.753439    9.540123
## 5 E3.25    8.924228    4.923580    4.629728    8.705340
## 6 E3.25   11.325952    4.068520    4.165692    8.696228
```

e.g. a expression matrix with each raw containing a gene expression for all samples

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Data.frame in R can be in:

wide format

```
##          X1420085_at X1418863_at X1425463_at X1416967_at
## 1 E3.25     3.027715    4.843137    5.500618    1.731217
## 2 E3.25     9.293016    5.530016    6.160900    9.697038
## 3 E3.25     2.940142    4.418059    4.584961    4.161240
## 4 E3.25     9.715243    5.982314    4.753439    9.540123
## 5 E3.25     8.924228    4.923580    4.629728    8.705340
## 6 E3.25    11.325952    4.068520    4.165692    8.696228
```

e.g. a expression matrix with each raw containing a gene expression for all samples

long format

```
##    sample       probe     value
## 1 1 E3.25 X1420085_at  3.027715
## 2 2 E3.25 X1420085_at  9.293016
## 3 3 E3.25 X1420085_at  2.940142
## 4 4 E3.25 X1420085_at  9.715243
## 5 5 E3.25 X1420085_at  8.924228
## 6 6 E3.25 X1420085_at 11.325952
```

e.g. a collapsed expression data with each row corresponding to a gene-sample pair

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Data.frame in R can be in:

wide format

```
##          X1420085_at X1418863_at X1425463_at X1416967_at
## 1 E3.25     3.027715    4.843137    5.500618    1.731217
## 2 E3.25     9.293016    5.530016    6.160900    9.697038
## 3 E3.25     2.940142    4.418059    4.584961    4.161240
## 4 E3.25     9.715243    5.982314    4.753439    9.540123
## 5 E3.25     8.924228    4.923580    4.629728    8.705340
## 6 E3.25    11.325952    4.068520    4.165692    8.696228
```

e.g. a expression matrix with each raw containing a gene expression for all samples

long format

```
##    sample       probe     value
## 1 1 E3.25 X1420085_at  3.027715
## 2 2 E3.25 X1420085_at  9.293016
## 3 3 E3.25 X1420085_at  2.940142
## 4 4 E3.25 X1420085_at  9.715243
## 5 5 E3.25 X1420085_at  8.924228
## 6 6 E3.25 X1420085_at 11.325952
```

e.g. a collapsed expression data with each row corresponding to a gene-sample pair

Each **row** corresponds to a **sample** and each **column** to a **feature** (or vice versa).

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Data.frame in R can be in:

wide format

```
##          X1420085_at X1418863_at X1425463_at X1416967_at
## 1 E3.25     3.027715    4.843137    5.500618    1.731217
## 2 E3.25     9.293016    5.530016    6.160900    9.697038
## 3 E3.25     2.940142    4.418059    4.584961    4.161240
## 4 E3.25     9.715243    5.982314    4.753439    9.540123
## 5 E3.25     8.924228    4.923580    4.629728    8.705340
## 6 E3.25    11.325952    4.068520    4.165692    8.696228
```

e.g. a expression matrix with each raw containing a gene expression for all samples

Each **row** corresponds to a **sample** and each **column** to a **feature** (or vice versa).

long format

```
##    sample       probe     value
## 1 1 E3.25 X1420085_at  3.027715
## 2 2 E3.25 X1420085_at  9.293016
## 3 3 E3.25 X1420085_at  2.940142
## 4 4 E3.25 X1420085_at  9.715243
## 5 5 E3.25 X1420085_at  8.924228
## 6 6 E3.25 X1420085_at 11.325952
```

e.g. a collapsed expression data with each row corresponding to a gene-sample pair

Feature and **sample** information is stored separately for each measurement in data columns.

Tidying data to use columns as aesthetics

```
ggplot( dftx,
  aes( x = X1426642_at, y = X1418765_at)) + geom_point() +
   facet_grid( Embryonic.day ~ lineage )
```

Data.frame in R can be in:

wide format

```
##          X1420085_at X1418863_at X1425463_at X1416967_at
## 1 E3.25     3.027715    4.843137    5.500618    1.731217
## 2 E3.25     9.293016    5.530016    6.160900    9.697038
## 3 E3.25     2.940142    4.418059    4.584961    4.161240
## 4 E3.25     9.715243    5.982314    4.753439    9.540123
## 5 E3.25     8.924228    4.923580    4.629728    8.705340
## 6 E3.25    11.325952    4.068520    4.165692    8.696228
```

e.g. a expression matrix with each raw containing a gene expression for all samples

Each **row** corresponds to a **sample** and each **column** to a **feature** (or vice versa).

long format

```
##    sample       probe     value
## 1 1 E3.25 X1420085_at  3.027715
## 2 2 E3.25 X1420085_at  9.293016
## 3 3 E3.25 X1420085_at  2.940142
## 4 4 E3.25 X1420085_at  9.715243
## 5 5 E3.25 X1420085_at  8.924228
## 6 6 E3.25 X1420085_at 11.325952
```

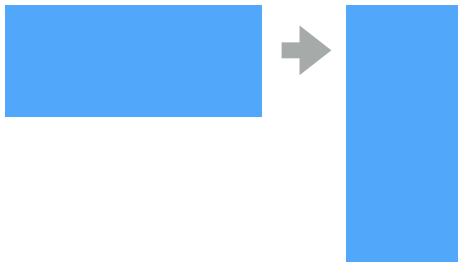
e.g. a collapsed expression data with each row corresponding to a gene-sample pair

Feature and **sample** information is stored separately for each measurement in data columns.

How do you switch :wide: <> :long: ?

`melt()` from wide to long

To convert between two data formats we can use functions **`melt()`**, and **`dcast()`** from package **`reshape2`**



```
head(genes_expression)
```

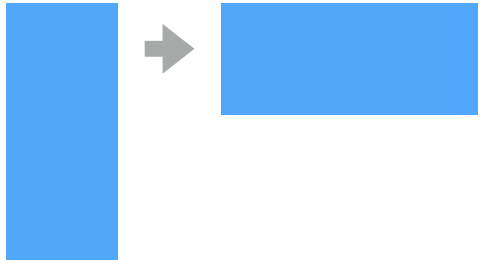
```
##         X1420085_at X1418863_at X1425463_at X1416967_at
## 1 E3.25    3.027715    4.843137    5.500618    1.731217
## 2 E3.25    9.293016    5.530016    6.160900    9.697038
## 3 E3.25    2.940142    4.418059    4.584961    4.161240
## 4 E3.25    9.715243    5.982314    4.753439    9.540123
## 5 E3.25    8.924228    4.923580    4.629728    8.705340
## 6 E3.25   11.325952    4.068520    4.165692    8.696228
```

```
library("reshape2")
genes = melt(genes_expression, varnames = c("sample", "probe"))
head(genes)
```

```
##    sample       probe     value
## 1 1 E3.25 X1420085_at  3.027715
## 2 2 E3.25 X1420085_at  9.293016
## 3 3 E3.25 X1420085_at  2.940142
## 4 4 E3.25 X1420085_at  9.715243
## 5 5 E3.25 X1420085_at  8.924228
## 6 6 E3.25 X1420085_at 11.325952
```

`dcast()` from long to wide

To convert between two data formats we can use functions **`melt()`**, and **`dcast()`** from package **`reshape2`**



```
head(genes)
```

```
##    sample       probe     value
## 1 1 E3.25 X1420085_at  3.027715
## 2 2 E3.25 X1420085_at  9.293016
## 3 3 E3.25 X1420085_at  2.940142
## 4 4 E3.25 X1420085_at  9.715243
## 5 5 E3.25 X1420085_at  8.924228
## 6 6 E3.25 X1420085_at 11.325952
```

```
wide <- dcast(genes, formula =  sample ~ probe, value.var = "value")
head(wide)
```

```
##    sample X1420085_at X1418863_at X1425463_at X1416967_at
## 1 1 E3.25    3.027715    4.843137    5.500618    1.731217
## 2 2 E3.25    9.293016    5.530016    6.160900    9.697038
## 3 3 E3.25    2.940142    4.418059    4.584961    4.161240
## 4 4 E3.25    9.715243    5.982314    4.753439    9.540123
## 5 5 E3.25    8.924228    4.923580    4.629728    8.705340
## 6 6 E3.25   11.325952    4.068520    4.165692    8.696228
```

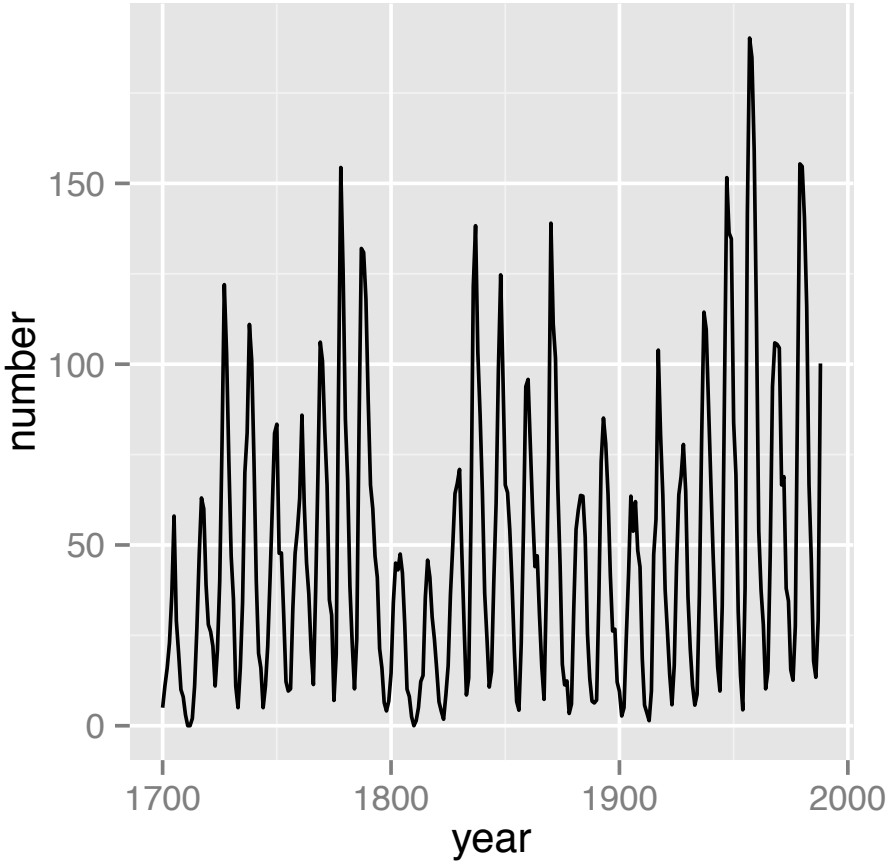
Choose **aspect ratio** so that banking = 45%

Yearly sunspot numbers 1849-1924 - changes in amplitude

Banking to 45 degrees:

Choose aspect ratio so that the median absolute slope is 1, i.e. at 45 degrees angle.

Sawtooth: Sunspot cycles typically rise more rapidly than they fall — steep rise and slow decline.



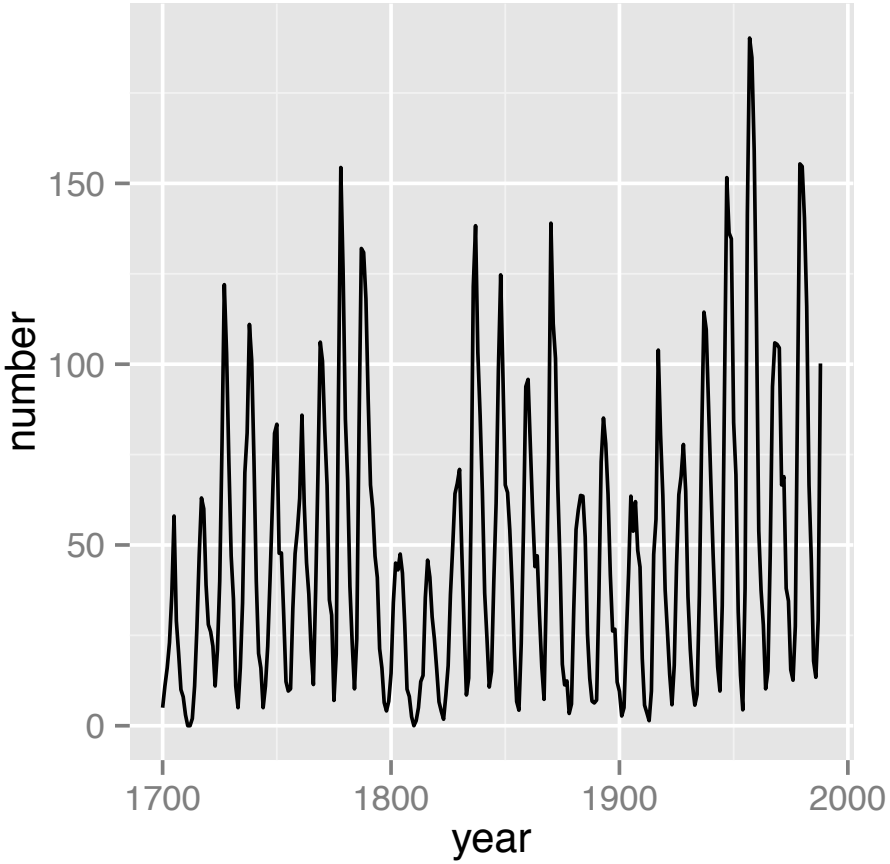
Choose **aspect ratio** so that banking = 45%

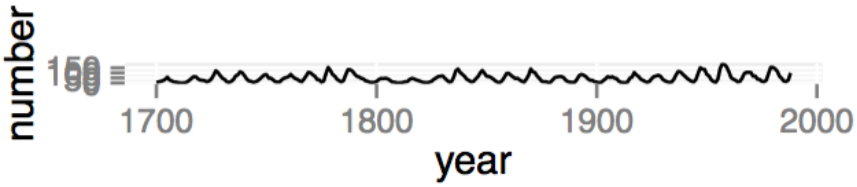
Yearly sunspot numbers 1849-1924 - changes in amplitude

Banking to 45 degrees:

Choose aspect ratio so that the median absolute slope is 1, i.e. at 45 degrees angle.

Sawtooth: Sunspot cycles typically rise more rapidly than they fall — steep rise and slow decline.





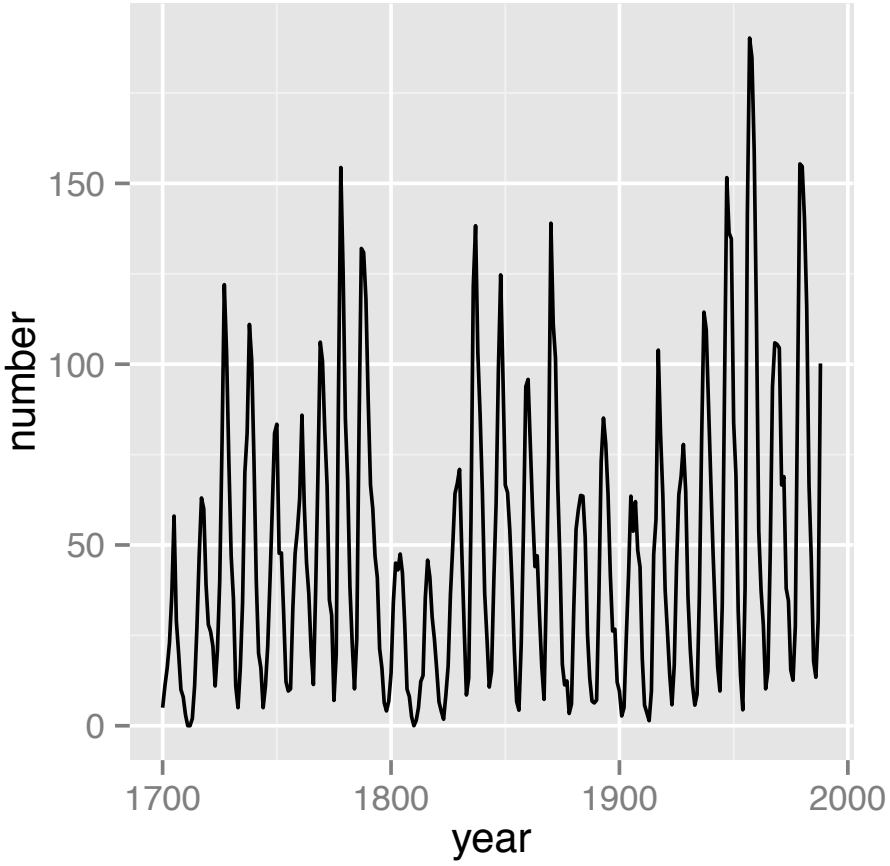
Choose **aspect ratio** so that banking = 45%

Yearly sunspot numbers 1849-1924 - changes in amplitude

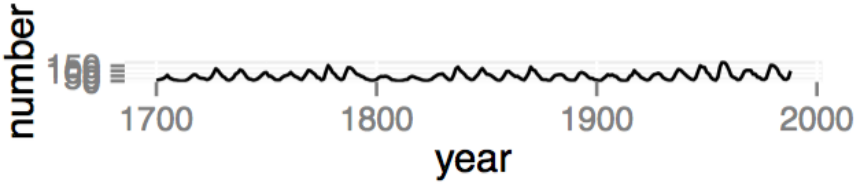
Banking to 45 degrees:

Choose aspect ratio so that the median absolute slope is 1, i.e. at 45 degrees angle.

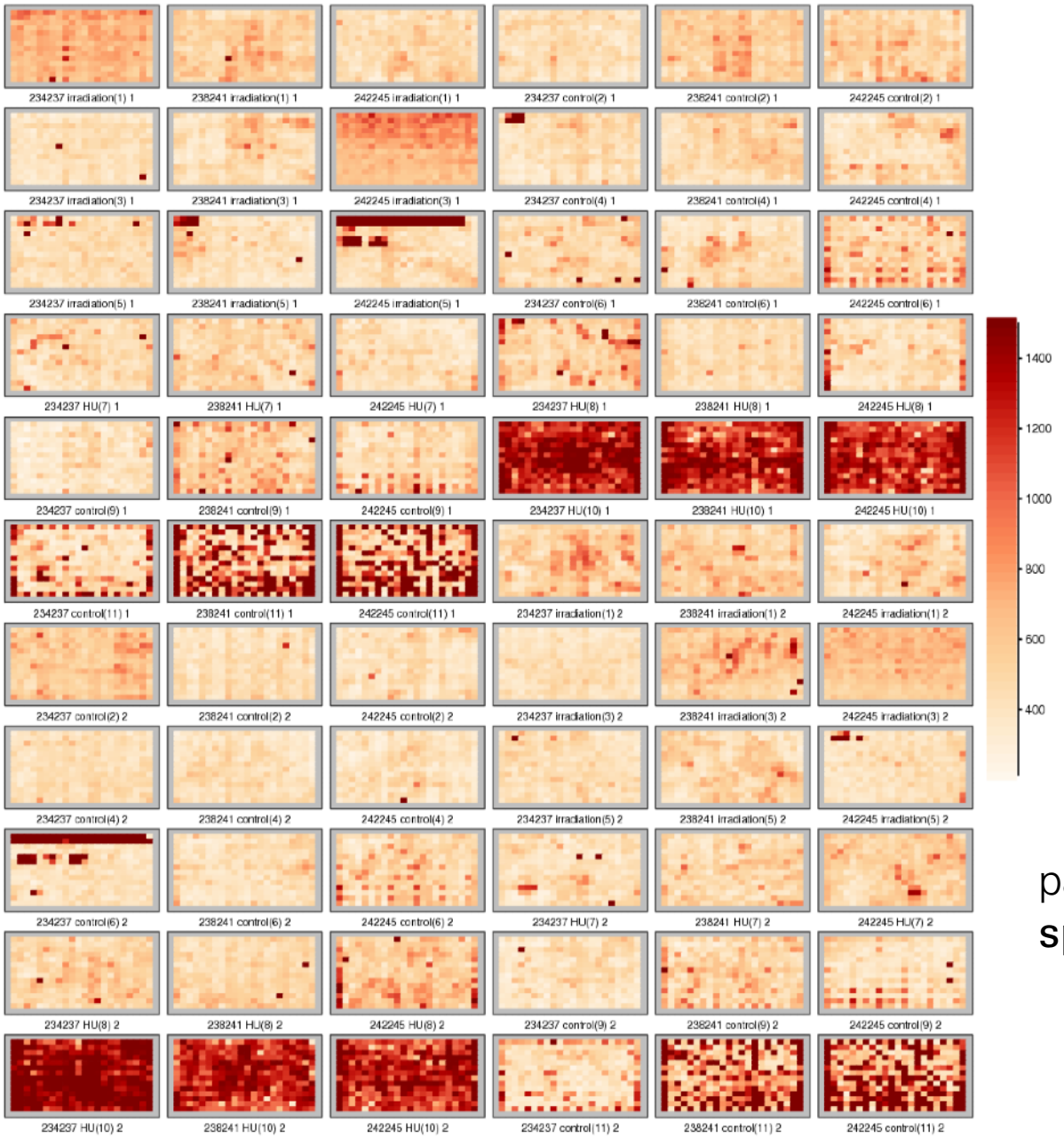
Sawtooth: Sunspot cycles typically rise more rapidly than they fall — steep rise and slow decline.



For plots where x- and y-axis have same units:
use 1:1 aspect ratio



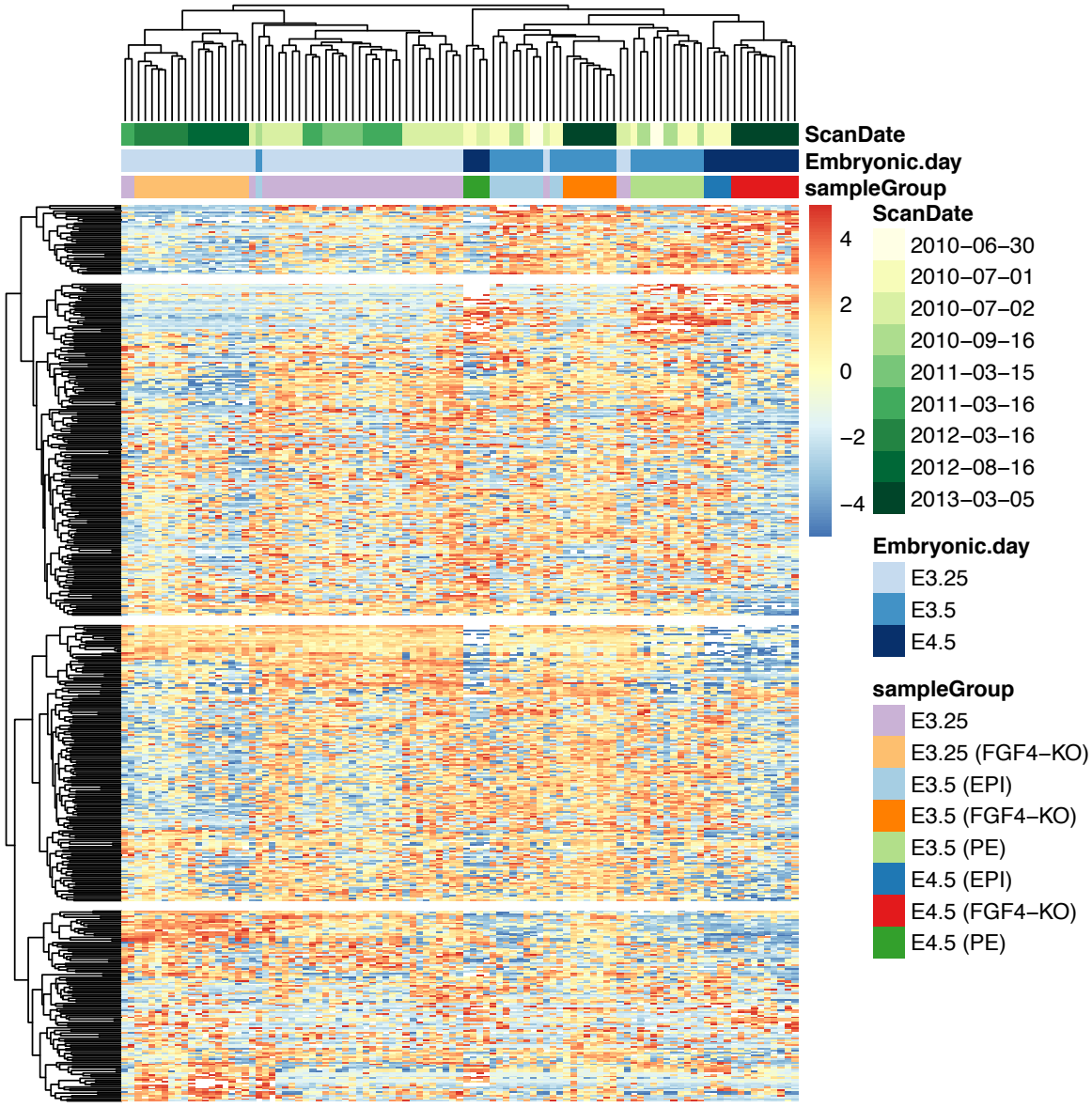
EDA for finding batch effects



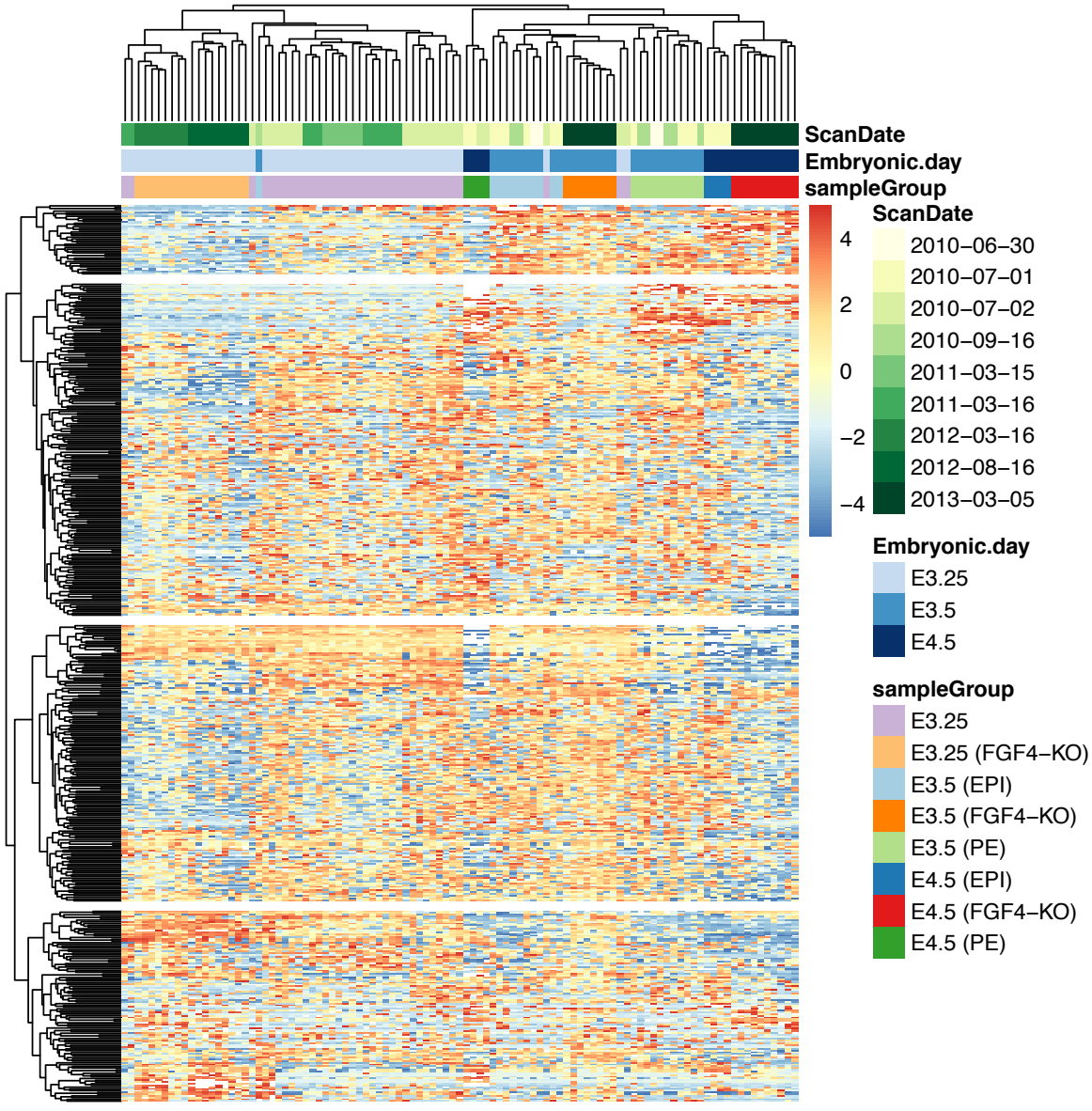
package
splots

Heatmaps for visualizing large matrices

Heatmaps for visualizing large matrices



Heatmaps for visualizing large matrices

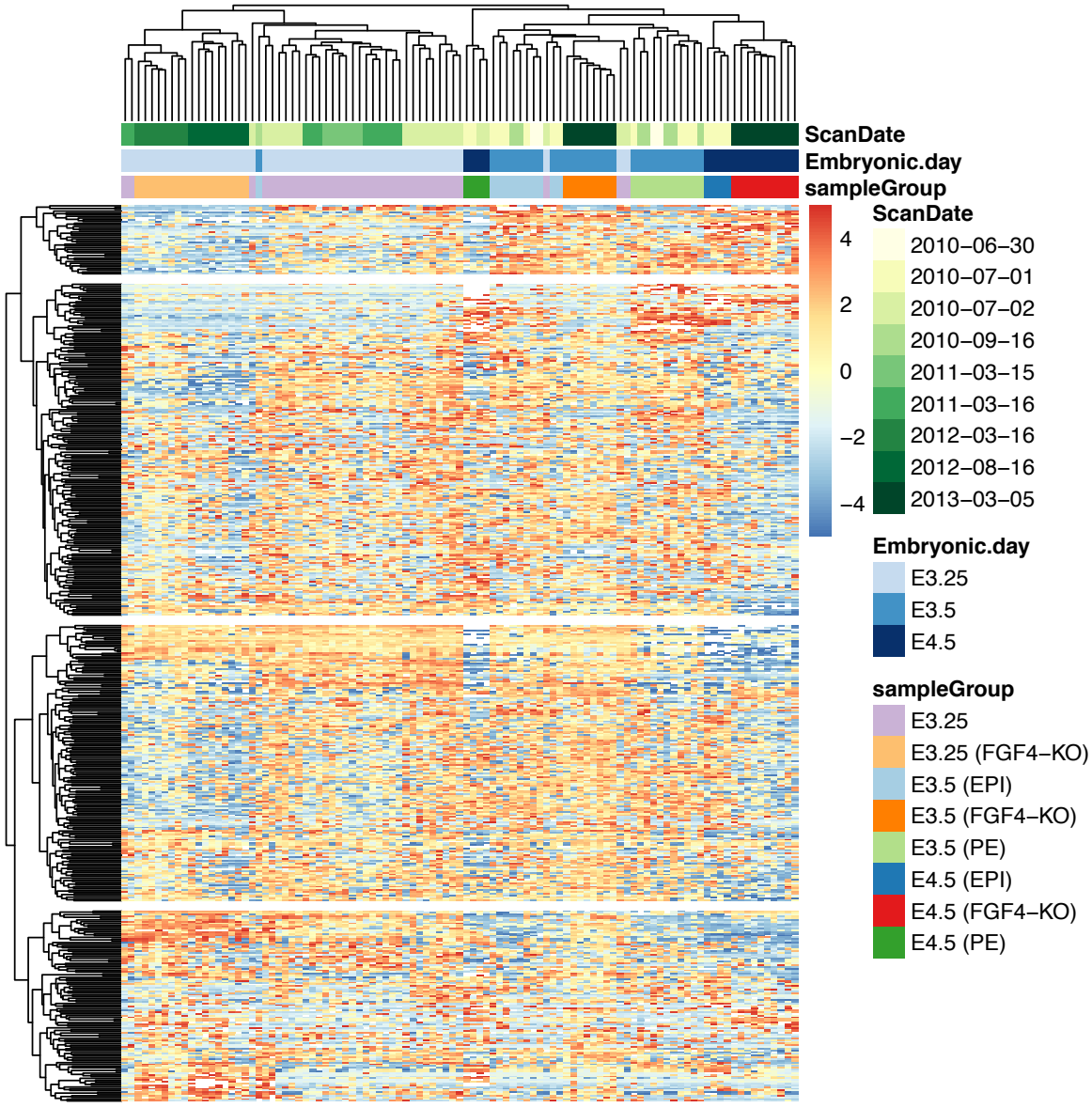


pheatmap

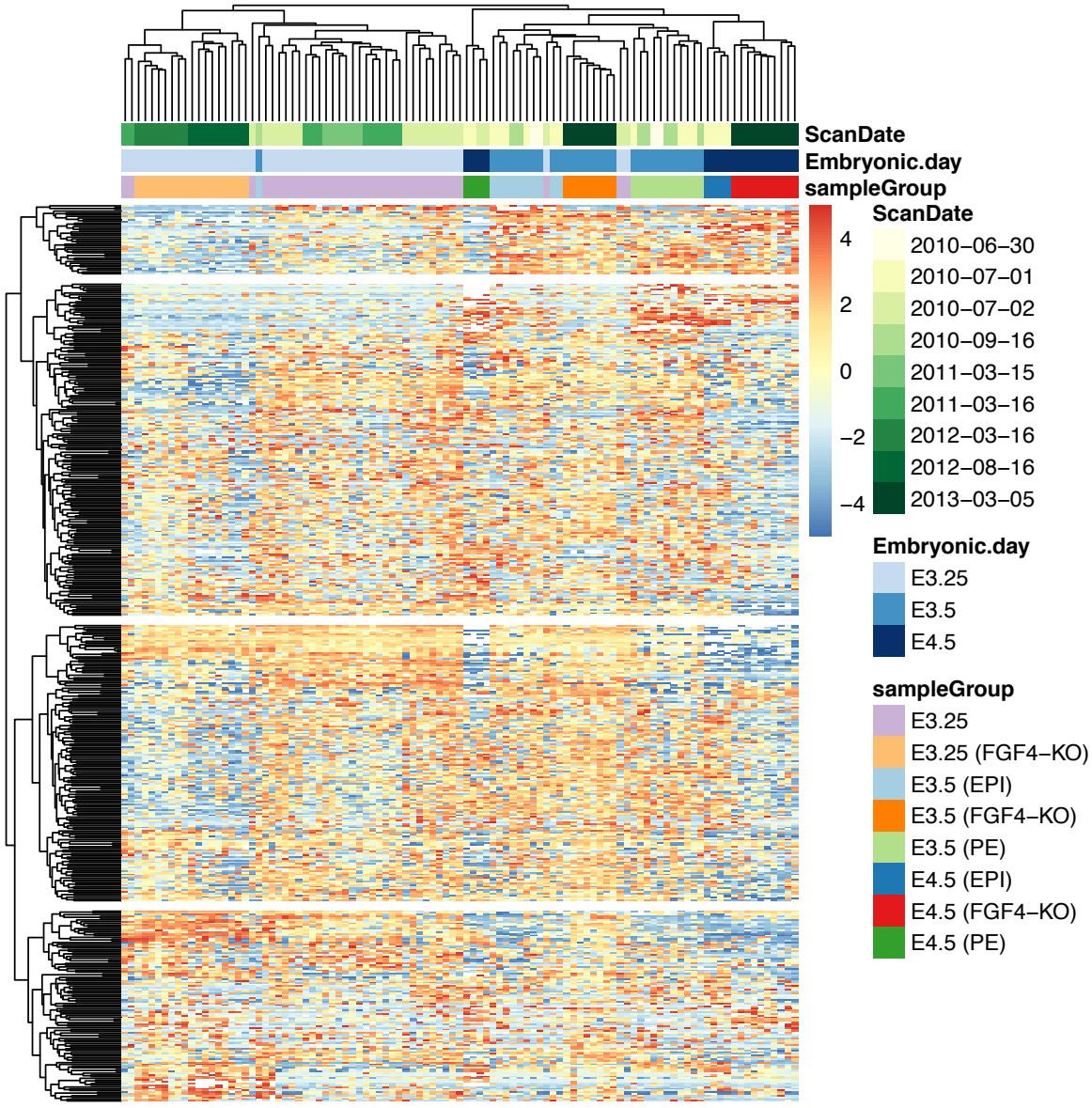
- many “reasonable” defaults
- easy to add column and row ‘metadata’ at the sides

See also *ComplexHeatmap* package

The **order** of dendrogram **branches** is not unique



The **order** of dendrogram **branches** is not unique



Goals for this lecture

1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

Interactivity

Use shiny or plotly
https://shiny.rstudio.com/gallery/genome-browser.html

Animations (time-dependent plots):
https://gganimate.com

Linked Charts
https://anders-biostat.github.io/linked-charts/

NB: ggvis is senescent

`plotly` interactive graphics

`plotly` interactive graphics

- `plotly` is a package for **visualization and a collaboration platform** for data science

`plotly` interactive graphics

- `plotly` is a package for **visualization and a collaboration platform** for data science
- produces **interactive graphics including 3D plots** (with zooming and rotating).

`plotly` interactive graphics

- `plotly` is a package for **visualization and a collaboration platform** for data science
- produces **interactive graphics including 3D plots** (with zooming and rotating).
- can **translate 'ggplot2' graphs to an interactive version**.

`plotly` interactive graphics

- `plotly` is a package for **visualization and a collaboration platform** for data science
- produces **interactive graphics including 3D plots** (with zooming and rotating).
- can **translate 'ggplot2' graphs to an interactive version**.
- You can open a **'plotly' account** to upload 'plotly' graphs and view or modify them in a web browser.

`plotly` interactive graphics

- `plotly` is a package for **visualization and a collaboration platform** for data science
- produces **interactive graphics including 3D plots** (with zooming and rotating).
- can **translate 'ggplot2' graphs to an interactive version**.
- You can open a **'plotly' account** to upload 'plotly' graphs and view or modify them in a web browser.

`plotly` interactive graphics

- `plotly` is a package for **visualization and a collaboration platform** for data science
- produces **interactive graphics including 3D plots** (with zooming and rotating).
- can **translate 'ggplot2' graphs to an interactive version**.
- You can open a **'plotly' account** to upload 'plotly' graphs and view or modify them in a web browser.

More on `plotly` can be found at https://plotly-book.cpsievert.me/

`plotly` interactive graphics demo

see 2019_Lec3_graphics_plots.html

Acknowledgements

Susan Holmes

Laura Marie J Symul

Hadley Wickham

Lan Huong Nguyen

Goals for this lecture

1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

Goals for this lecture

1. Discuss the principles of **good *vs* bad** data viz
2. Review base R plotting
3. Understand the **grammar of graphics** concept
4. Introduce, explain and use the `ggplot()` function
5. Discuss how to plot 1D, 2D, 3-5D data and select the most appropriate plot type. Use faceting
6. Use visualization for the inspection of large datasets and discovery of global trends (e.g. batch effects)
7. Implement interactive (3D) visualization

What is your main take-away from each section?

Questions