Visualization and Data Structure

Principles of graph grammar and tidy data

Thiago de Paula Oliveira
thiago.oliveira@ed.ac.uk
https://prof-thiagooliveira.netlify.com

20th September 2021

@HighlanderLab

Contents

How install, load, and update R-packages

Installing packages

- **CRAN**:

```
# install the ggplot2 package
R> install.packages("ggplot2")
R> install.packages(c("dplyr", "AlphaSimR"))
```

- **GitHub**:

```
# install the ggplot2 package from GitHub Repo
# install.packages("devtools")
R> devtools::install_github("tidyverse/ggplot2")
```

Removing a package

```
R> remove.packages("ggplot2")
```

How install, load, and update R-packages

Installing packages

- **CRAN**:

```
# install the ggplot2 package
R> install.packages("ggplot2")
R> install.packages(c("dplyr", "AlphaSimR"))
```

- **GitHub**:

```
# install the ggplot2 package from GitHub Repo
# install.packages("devtools")
R> devtools::install_github("tidyverse/ggplot2")
```

Removing a package

```
R> remove.packages("ggplot2")
```

How install, load, and update R-packages

Installing packages

- **CRAN**:

```
# install the ggplot2 package
R> install.packages("ggplot2")
R> install.packages(c("dplyr", "AlphaSimR"))
```

- **GitHub**:

```
# install the ggplot2 package from GitHub Repo
# install.packages("devtools")
R> devtools::install_github("tidyverse/ggplot2")
```

Removing a package

```
R> remove.packages("ggplot2")
```

How install, load, and update R-packages

Loading packages

- One package:

```
R > library("ggplot2")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+            "dplyr")
R> invisible(lapply(pkg, library,
+             character.only = TRUE))
```

- Unloading a package:

```
R> detach("ggplot2", unload = TRUE)
```

Updating packages

- One package:

```
R> update.package("AlphaSimR")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+            "dplyr")
R> update.packages(oldPkgs = pkg)
```

- All packages:

```
R> update.packages(ask=FALSE)
```

How install, load, and update R-packages

Loading packages

- One package:

```
R > library("ggplot2")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> invisible(lapply(pkg, library,
+            character.only = TRUE))
```

- Unloading a package:

```
R> detach("ggplot2", unload = TRUE)
```

Updating packages

- One package:

```
R> update.package("AlphaSimR")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> update.packages(oldPkgs = pkg)
```

- All packages:

```
R> update.packages(ask=FALSE)
```

How install, load, and update R-packages

Loading packages

- One package:

```
R > library("ggplot2")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> invisible(lapply(pkg, library,
+            character.only = TRUE))
```

- Unloading a package:

```
R> detach("ggplot2", unload = TRUE)
```

Updating packages

- One package:

```
R> update.package("AlphaSimR")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> update.packages(oldPkgs = pkg)
```

- All packages:

```
R> update.packages(ask=FALSE)
```

How install, load, and update R-packages

Loading packages

- One package:

```
R > library("ggplot2")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> invisible(lapply(pkg, library,
+            character.only = TRUE))
```

- Unloading a package:

```
R> detach("ggplot2", unload = TRUE)
```

Updating packages

- One package:

```
R> update.package("AlphaSimR")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> update.packages(oldPkgs = pkg)
```

- All packages:

```
R> update.packages(ask=FALSE)
```

How install, load, and update R-packages

Loading packages

- One package:

```
R > library("ggplot2")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> invisible(lapply(pkg, library,
+            character.only = TRUE))
```

- Unloading a package:

```
R> detach("ggplot2", unload = TRUE)
```

Updating packages

- One package:

```
R> update.package("AlphaSimR")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> update.packages(oldPkgs = pkg)
```

- All packages:

```
R> update.packages(ask=FALSE)
```

How install, load, and update R-packages

Loading packages

- One package:

```
R > library("ggplot2")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> invisible(lapply(pkg, library,
+            character.only = TRUE))
```

- Unloading a package:

```
R> detach("ggplot2", unload = TRUE)
```

Updating packages

- One package:

```
R> update.package("AlphaSimR")
```

- Multiple packages at once:

```
R> pkg <- c("ggplot2", "AlphaSimR",
+           "dplyr")
R> update.packages(oldPkgs = pkg)
```

- All packages:

```
R> update.packages(ask=FALSE)
```

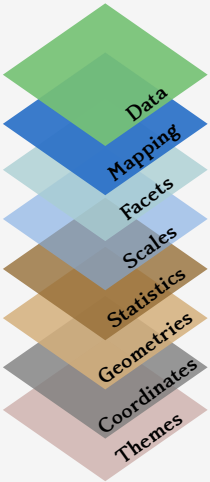
ggplot2
The Grammar of Graphics

The Grammar of Graphics

- **What is grammar of graphics?**
  - a tool that enables us to concisely describe the components of a graphic
- Provides a strong foundation for understanding a **diverse range of graphics**
- The layered grammar defines the **components of a plot**
- The power of the grammar shall be illustrated over the course

Wickham, H. (2010)

Decompose Graphics into its components

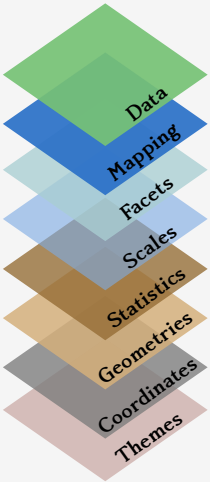


The Grammar of Graphics

- **What is grammar of graphics?**
  - a tool that enables us to concisely describe the components of a graphic
- Provides a strong foundation for understanding a **diverse range of graphics**
- The layered grammar defines the **components of a plot**
- The power of the grammar shall be illustrated over the course

Wickham, H. (2010)

Decompose Graphics into its components

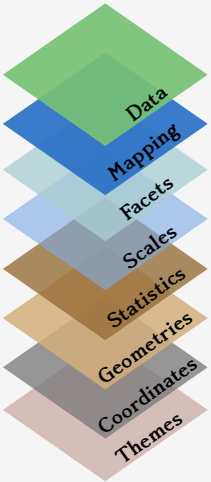


The Grammar of Graphics

- **What is grammar of graphics?**
  - a tool that enables us to concisely describe the components of a graphic
- Provides a strong foundation for understanding a **diverse range of graphics**
- The layered grammar defines the **components of a plot**
- The power of the grammar shall be illustrated over the course

Wickham, H. (2010)

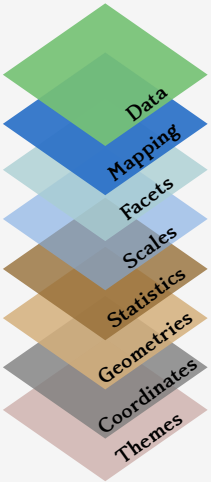
Decompose Graphics into its components



The Grammar of Graphics

- **What is grammar of graphics?**
  - a tool that enables us to concisely describe the components of a graphic
- Provides a strong foundation for understanding a **diverse range of graphics**
- The layered grammar defines the **components of a plot**
- The power of the grammar shall be illustrated over the course

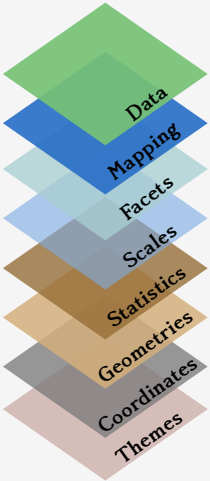
Decompose Graphics into its components

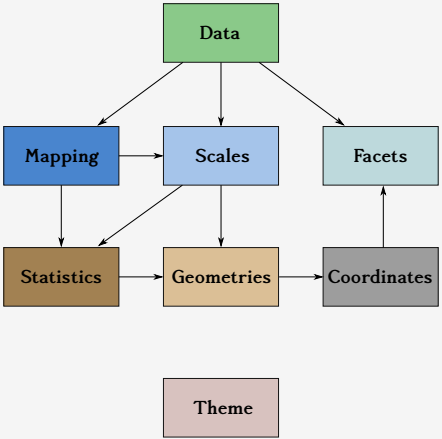


Wickham, H. (2010)

The Grammar of Graphics

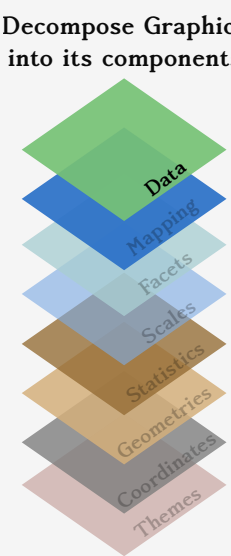
Decompose Graphics into its components





Data

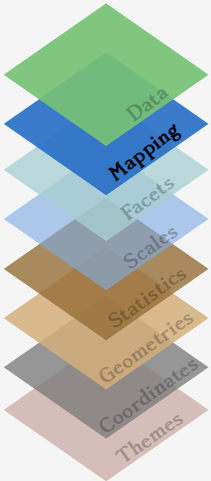
- Data turns an abstract graphic into a **concrete graphic**!
- Render these data to create the graphical object:
  - tidy data is required!
- Must be stored as an R data frame



Mapping

- **Aesthetic mapping**: map data to visual elements or parameters
  - processed by scales and statistics layers
  - used by graphical properties in the **geometry**
- **aesthetics** in `ggplot2`: `aes()` describe **visual characteristics:**
  - position, size, color, shape, fill, transparency, ...
- **Facet mapping**: link variables in the data to panels:
  - `facet_*`

Decompose Graphics into its components



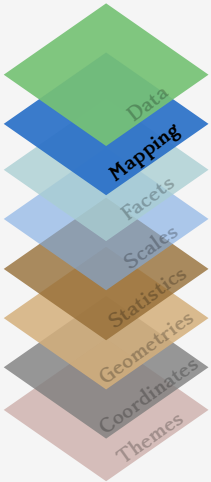
Mapping - adding aes()

```
R> cbp <- readRDS("animal_sim.RDS")
R> p <- ggplot(data = cbp, aes(x = Year, y = Pheno,
+               color = Sex))
```

Ind	Father	Mother	Year	Sex	Herd	Pheno
1	NA	NA	0	M	E	37.67
2	NA	NA	0	M	B	35.80
3	NA	NA	0	M	A	28.43
4	NA	NA	0	M	D	33.63
⋮	⋮	⋮	⋮	⋮	⋮	
9997	8342	8677	9	F	A	58.43
9998	8088	8510	9	F	E	58.17
9999	8449	8685	9	F	D	56.72
10000	8296	8854	9	F	B	67.08

- Plot object p cannot be displayed: we need at least one layer

Decompose Graphics into its components



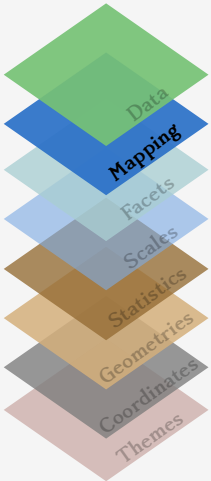
Mapping - adding aes()

```
R> cbp <- readRDS("animal_sim.RDS")
R> p <- ggplot(data = cbp, aes(x = Year, y = Pheno,
+               color = Sex))
```

Ind	Father	Mother	Year	Sex	Herd	Pheno
1	NA	NA	0	M	E	37.67
2	NA	NA	0	M	B	35.80
3	NA	NA	0	M	A	28.43
4	NA	NA	0	M	D	33.63
⋮	⋮	⋮	⋮	⋮	⋮	
9997	8342	8677	9	F	A	58.43
9998	8088	8510	9	F	E	58.17
9999	8449	8685	9	F	D	56.72
10000	8296	8854	9	F	B	67.08

- Plot object p cannot be displayed: we need at least one layer

Decompose Graphics into its components

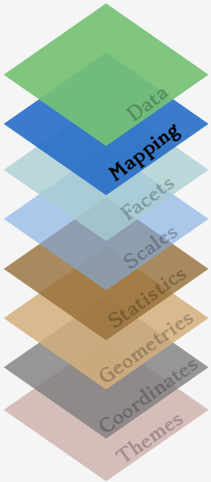


Mapping - adding the first layer

- Every `ggplot2::layer()` specifies a **geom** or a **stat** or **both**
- '+' is used to combine `ggplot2` components

```
R> p <- ggplot(data = cbp, aes(x = Year, y = Pheno,
+               color = Sex))
R> p  + layer(geom = "point", stat = "identity",
+             position = "identity",
+             params = list(shape=1, na.rm = FALSE))
```

Decompose Graphics into its components

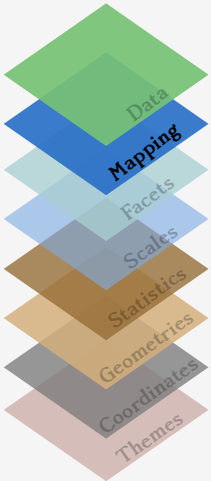


Mapping - adding the first layer

- Every `ggplot2::layer()` specifies a **geom** or a **stat** or **both**
- '+' is used to combine `ggplot2` components

```
R> p <- ggplot(data = cbp, aes(x = Year, y = Pheno,
+              color = Sex))
R> p  + layer(geom = "point", stat = "identity",
+             position = "identity",
+             params = list(shape=1, na.rm = FALSE))
```

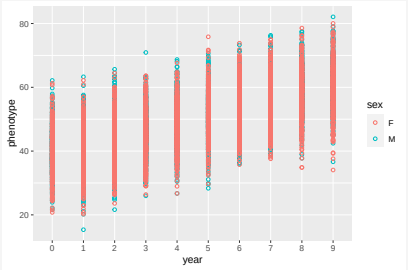
Decompose Graphics into its components



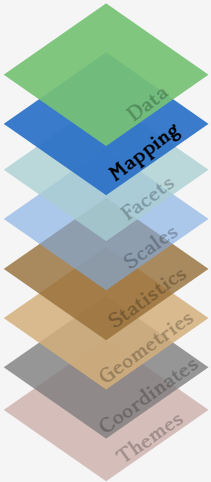
Mapping - adding the first layer

- Every `ggplot2::layer()` specifies a **geom** or a **stat** or **both**
- '+' is used to combine `ggplot2` components

```
R> p <- ggplot(data = cbp, aes(x = Year, y = Pheno,
+               color = Sex))
R> p  + layer(geom = "point", stat = "identity",
+             position = "identity",
+             params = list(shape=1, na.rm = FALSE))
```



Decompose Graphics into its components



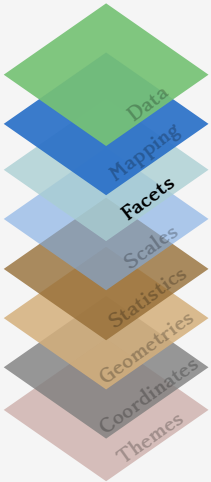
Facets

- **Defines the number of panels** based on the number of subsets
- The same logic defined for the components are applied for each panel
- **Facets** should be applied when the subset **carries a meaning**

Type of facets

- `facet_null()`: a **single plot**, the default.
- `facet_wrap()`: '**wraps**' **a 1D** ribbon of panels **into 2D**
- `facet_grid()`: **produces a 2D grid** of panels defined by **variables** which form the rows and columns

Decompose Graphics into its components

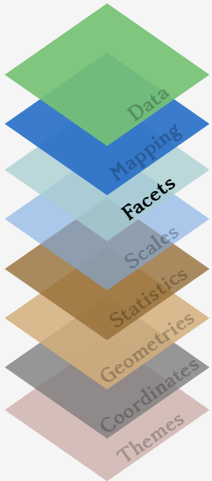


Facets

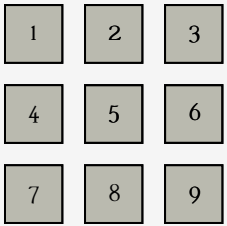
- **Defines the number of panels** based on the number of subsets
- The same logic defined for the components are applied for each panel
- **Facets** should be applied when the subset **carries a meaning**

Type of facets

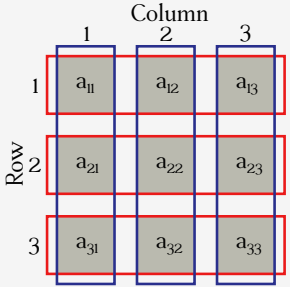
- `facet_null()`: a **single plot**, the default.
- `facet_wrap()`: **'wraps' a 1D** ribbon of panels **into 2D**
- `facet_grid()`: **produces a 2D grid** of panels defined by **variables** which form the rows and columns



Facets

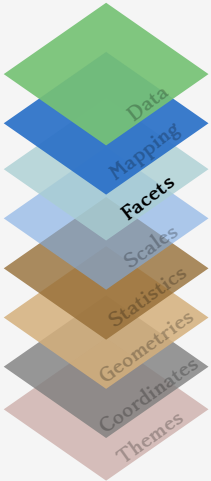



```
facet_wrap(
  facets,
  nrow = NULL,
  ncol = NULL,
  scales = "fixed",
  shrink = TRUE,
  labeller = "label_value",
  as.table = TRUE,
  switch = NULL,
  drop = TRUE,
  dir = "h",
  strip.position = "top"
)
```



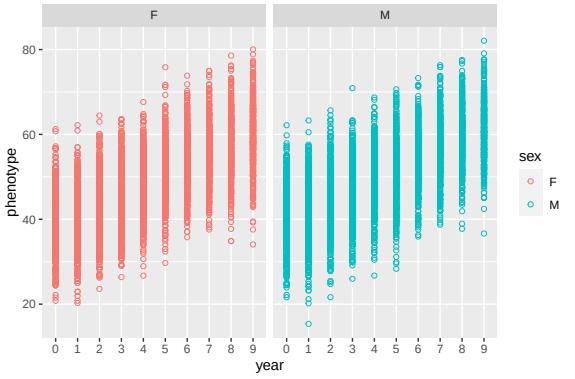

```
facet_grid(
  rows = NULL,
  cols = NULL,
  scales = "fixed",
  space = "fixed",
  shrink = TRUE,
  labeller = "label_value",
  as.table = TRUE,
  switch = NULL,
  drop = TRUE,
  margins = FALSE,
  facets = NULL
)
```

Decompose Graphics into its components

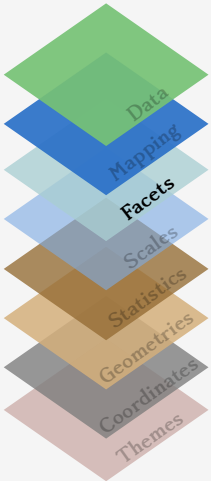


Facets - adding a wrap

```
R> p + layer(geom = "point", stat = "identity",
+            position = "identity",
+            params = list(shape=1, na.rm = FALSE)) +
+      facet_wrap(facet = "sex")
```

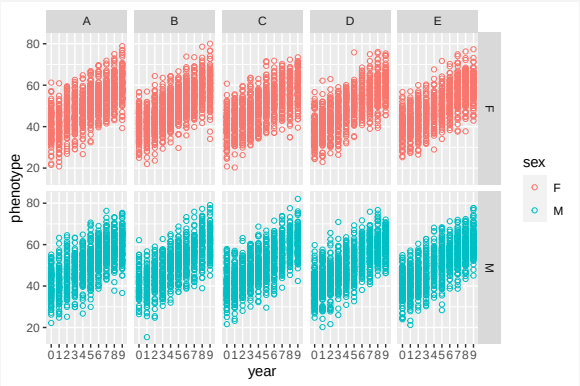


Decompose Graphics into its components

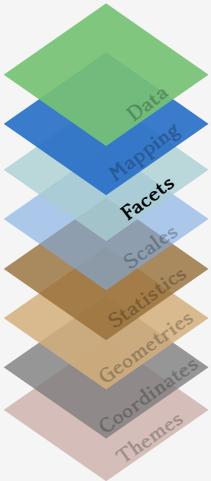


Facets - adding a grid

```
1 R> p + layer(geom = "point", stat = "identity",
2 +            position = "identity",
3 +            params = list(shape=1, na.rm = FALSE)) +
4 +      facet_grid(rows = vars(sex), cols = vars(herd))
```



Decompose Graphics into its components

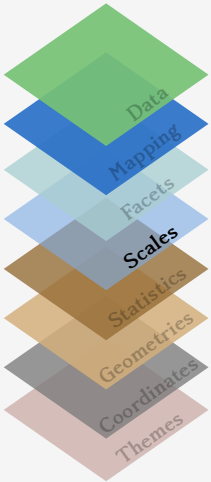


Scales

- **Control the mapping** from data to aesthetics
  - size, colour, position, shape, ...

- Provide the tools that let you interpret the plot
  - axes: every plot has **two position scales corresponding to the x and y** aesthetics!
    - **numeric**: `scale_x_continuous()` and `scale_y_continuous()`
    - **discrete**: `scale_x_discrete()` and `scale_y_discrete()`
    - **binned**: `scale_x_binned()` and `scale_y_binned()`
    - **date**: `scale_x_date()` and `scale_y_date()`
  - legends
    - **fill**: `scale_fill_gray()`; `scale_fill_hue()`; `scale_fill_manual()`
    - **colour discrete**: `scale_colour_brewer()`
    - **colour continuous**: `scale_colour_distiller()`
    - **colour binned**: `scale_colour_fermenter()`

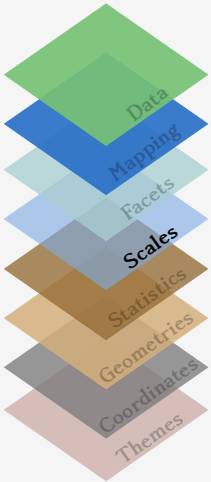
Decompose Graphics into its components



Scales

- **Control the mapping** from data to aesthetics
  - size, colour, position, shape, ...
- Provide the tools that let you interpret the plot
  - axes: every plot has **two position scales corresponding to the x and y** aesthetics!
    - **numeric**: `scale_x_continuous()` and `scale_y_continuous()`
    - **discrete**: `scale_x_discrete()` and `scale_y_discrete()`
    - **binned**: `scale_x_binned()` and `scale_y_binned()`
    - **date**: `scale_x_date()` and `scale_y_date()`
  - legends
    - **fill**: `scale_fill_gray()`; `scale_fill_hue()`; `scale_fill_manual()`
    - **colour discrete**: `scale_colour_brewer()`
    - **colour continuous**: `scale_colour_distiller()`
    - **colour binned**: `scale_colour_fermenter()`

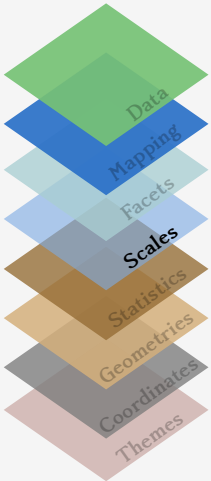
Decompose Graphics into its components



Scales

- **Control the mapping** from data to aesthetics
  - size, colour, position, shape, ...
- Provide the tools that let you interpret the plot
  - axes: every plot has **two position scales corresponding to the x and y** aesthetics!
    - **numeric**: `scale_x_continuous()` and `scale_y_continuous()`
    - **discrete**: `scale_x_discrete()` and `scale_y_discrete()`
    - **binned**: `scale_x_binned()` and `scale_y_binned()`
    - **date**: `scale_x_date()` and `scale_y_date()`
  - legends
    - **fill**: `scale_fill_gray()`; `scale_fill_hue()`; `scale_fill_manual()`
    - **colour discrete**: `scale_colour_brewer()`
    - **colour continuous**: `scale_colour_distiller()`
    - **colour binned**: `scale_colour_fermenter()`

Decompose Graphics into its components



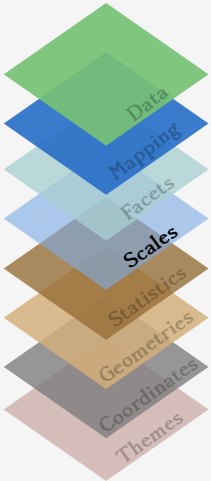
Scales

- You don't need to know how scales work to make plots with `ggplot2`

Changing colour

```
R> (p2 <- p + layer(geom = "point", stat = "identity",
+                   position = "identity",
+                   params = list(shape=1, na.rm = FALSE)) +
+                   facet_grid(rows = vars(sex), cols = vars(herd)) +
+             scale_colour_manual(values = c("#00AFBB", "#FC4E07")))
R> (p3 <- p2 + scale_color_brewer(palette = "Dark2"))
```

Decompose Graphics into its components

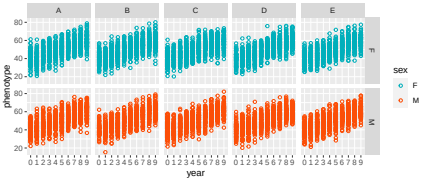


Scales

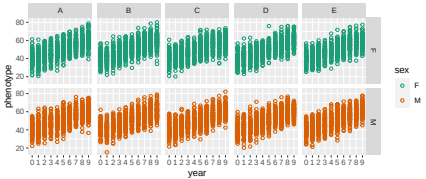
- You don't need to know how scales work to make plots with `ggplot2`

Changing colour

```
R> (p2 <- p + layer(geom = "point", stat = "identity",
+                   position = "identity",
+                   params = list(shape=1, na.rm = FALSE)) +
+                   facet_grid(rows = vars(sex), cols = vars(herd)) +
+             scale_colour_manual(values = c("#00AFBB", "#FC4E07")))
R> (p3 <- p2 + scale_color_brewer(palette = "Dark2"))
```

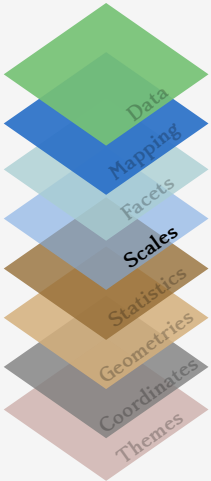


(c) plot p2



(d) plot p3

Decompose Graphics into its components



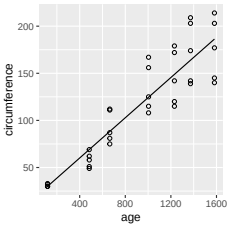
Statistics

- **Statistical transformation** to use on the data for specified layer
  - Count the number of observations in each category (bar chart)
  - Add regression line
  - Calculate summary statistics (boxplot)

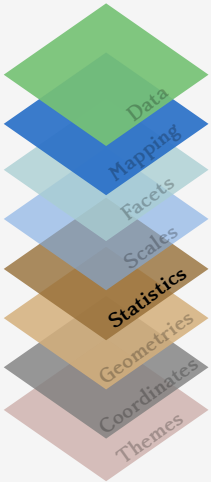
Linear regression $E(y_{ij}) = \mu + \beta \text{Age}_{ij}$

```
Rows: 35
Columns: 3
$ Tree          <ord> 1, 1, 1, ..., 5, 5
$ age           <dbl> 118, 484, 664, ..., 1372, 1572
$ circumference <dbl> 30, 58, 87, ..., 174, 177
```

```
1  R> data(Orange) # package datasets
2  R> ggplot(data = Orange,
3  +         aes(x = age, y = circumference)) +
4  +    layer(geom = "point", stat = "identity",
5  +          position = "identity",
6  +          params = list(shape=1, na.rm = FALSE)) +
7  +    layer(geom = "line", stat = "smooth",
8  +          position = "identity",
9  +          params = list(method = "lm", se = FALSE))
```

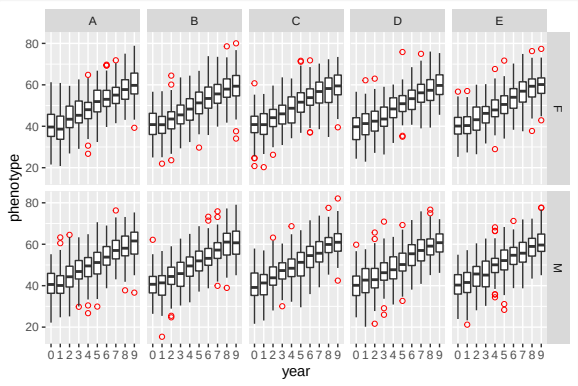


Decompose Graphics into its components

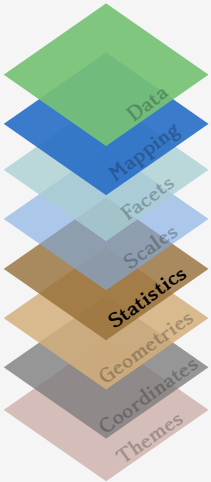


Statistics - boxplot

```
R> ggplot(data = cbp, aes(x = year, y = phenotype, group = year)) +
+    layer(geom = "boxplot", stat = "boxplot", position = "identity",
+          params = list(outlier.colour="red", outlier.shape = 1,
+                        na.rm = FALSE))
+    facet_grid(rows = vars(sex), cols = vars(herd))
```



Decompose Graphics into its components



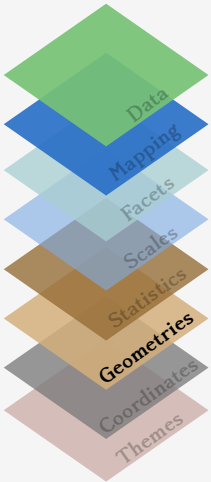
Geometries

- The way as `ggplot2` **interpret aesthetics** as graphical representation
- Geometry corresponds to the **type of graphic** (histogram, box plot, density plot, dot plot, line plot, polygons, ...)

Linear Regression: $E(y_{ij}) = \mu_i + \beta_i \text{Age}_{ij}$

```
R> ggplot(data = Orange, aes(x = age, y = circumference)) +
+   layer(geom = "point", stat = "identity", position = "identity",
+         params = list(shape=1, na.rm = FALSE)) +
+   layer(geom = "line", stat = "smooth", position = "identity",
+   params = list(method = "lm", se = FALSE)) + facet_wrap(facet = "Tree")
```

Decompose Graphics into its components

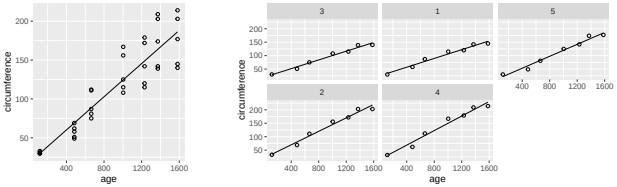


Geometries

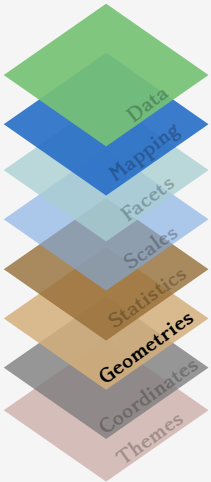
- The way as `ggplot2` **interpret aesthetics** as graphical representation
- Geometry corresponds to the **type of graphic** (histogram, box plot, density plot, dot plot, line plot, polygons, ...)

Linear Regression: $E(y_{ij}) = \mu_i + \beta_i \text{Age}_{ij}$

```
1  R> ggplot(data = Orange, aes(x = age, y = circumference)) +
2  +   layer(geom = "point", stat = "identity", position = "identity",
3  +         params = list(shape=1, na.rm = FALSE)) +
4  +   layer(geom = "line", stat = "smooth", position = "identity",
5  +   params = list(method = "lm", se = FALSE)) + facet_wrap(facet = "Tree")
```



Decompose Graphics into its components



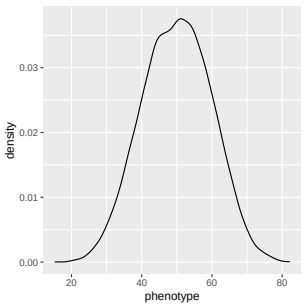
Shortcut Functions: `stat_*()` and `geom_*()`

- we can use `geom_*()` and `stat_*()` shortcut functions rather than `layer()`
- each `geom_*()` has a default `stat_*()` (**the opposite is also true**)
- `help.search("^geom_", package = "ggplot2")`

ggplot2::geom_abline	Reference lines: horizontal, vertical, and diagonal
ggplot2::geom_bar	Bar charts
ggplot2::geom_bin_2d	Heatmap of 2d bin counts
ggplot2::geom_blank	Draw nothing
ggplot2::geom_boxplot	A box and whiskers plot (in the style of Tukey)
ggplot2::geom_contour	2D contours of a 3D surface
ggplot2::geom_count	Count overlapping points
ggplot2::geom_density	Smoothed density estimates
ggplot2::geom_density_2d	Contours of a 2D density estimate
ggplot2::geom_dotplot	Dot plot
ggplot2::geom_errorbarh	Horizontal error bars
ggplot2::geom_function	Draw a function as a continuous curve
ggplot2::geom_hex	Hexagonal heatmap of 2d bin counts
ggplot2::geom_freqpoly	Histograms and frequency polygons
ggplot2::geom_jitter	Jittered points
ggplot2::geom_crossbar	Vertical intervals: lines, crossbars & errorbars
ggplot2::geom_map	Polygons from a reference map
ggplot2::geom_path	Connect observations
ggplot2::geom_point	Points
ggplot2::geom_polygon	Polygons
ggplot2::geom_qq_line	A quantile-quantile plot
ggplot2::geom_quantile	Quantile regression
ggplot2::geom_ribbon	Ribbons and area plots
ggplot2::geom_rug	Rug plots in the margins
ggplot2::geom_segment	Line segments and curves
ggplot2::geom_smooth	Smoothed conditional means
ggplot2::geom_spoke	Line segments parameterised by location, direction and distance
ggplot2::geom_label	Text
ggplot2::geom_raster	Rectangles
ggplot2::geom_violin	Violin plot
ggplot2::CoordSf	Visualise sf objects

Density plot: geom_density()

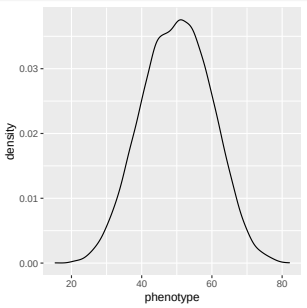
```
R> p <- ggplot(data = cbp)
R> p + geom_density(aes(x=phenotype))
```




```
R> library(ggridges)
R> p <- ggplot(data = cbp)
R> p +
+  geom_density_ridges(aes(x = phenotype, y = year,
+                          fill = year))
```

Density plot: geom_density()

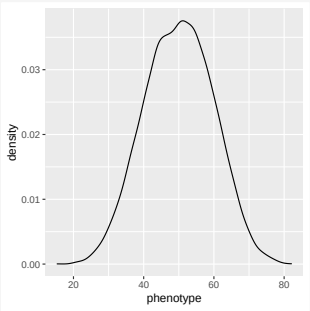
```
R> p <- ggplot(data = cbp)
R> p + geom_density(aes(x=phenotype))
```



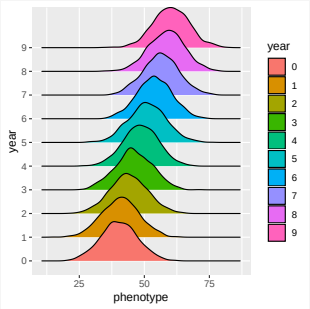

```
R> library(ggridges)
R> p <- ggplot(data = cbp)
R> p +
+  geom_density_ridges(aes(x = phenotype, y = year,
+                          fill = year))
```

Density plot: geom_density()

```
R> p <- ggplot(data = cbp)
R> p + geom_density(aes(x=phenotype))
```

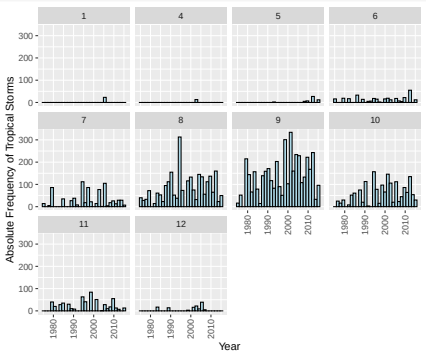



```
R> library(ggridges)
R> p <- ggplot(data = cbp)
R> p +
+  geom_density_ridges(aes(x = phenotype, y = year,
+                          fill = year))
```



Histogram: geom_histogram()

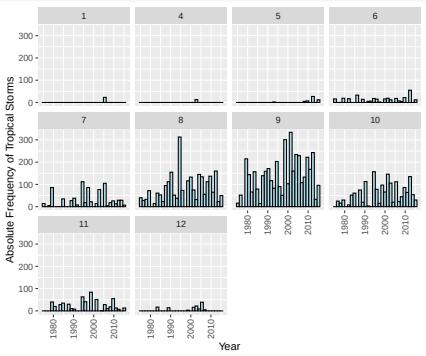
```
R> ggplot(data = storms, aes(x = year)) +
+   facet_wrap(~month)+
+   geom_histogram(fill = "lightblue",
+                  color = "black")+
+   ylab("Absolute Frequency") + xlab("Year") +
+   theme(axis.text.x = element_text(angle = 90,
                                    vjust = 0.5, hjust=1))
```




```
R> ggplot(data = storms, aes(x = year)) +
+   geom_histogram(aes(y=stat(count)/sum(..count..)),
+   fill = "lightblue", color = "black")+
+   scale_y_continuous(labels=scales::percent) +
+   ylab("Relative Frequency") + xlab("Year") +
+   theme(axis.text.x = element_text(angle = 90,
                                    vjust = 0.5, hjust=1))
```

Histogram: geom_histogram()

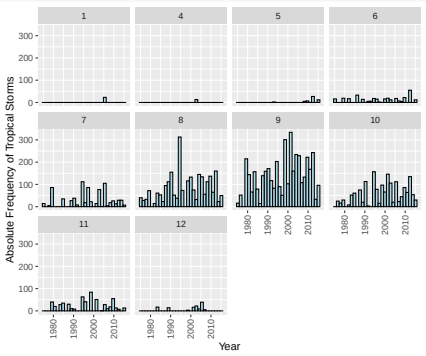
```
R> ggplot(data = storms, aes(x = year)) +
+   facet_wrap(~month)+
+   geom_histogram(fill = "lightblue",
+                  color = "black")+
+   ylab("Absolute Frequency") + xlab("Year") +
+   theme(axis.text.x = element_text(angle = 90,
                                    vjust = 0.5, hjust=1))
```



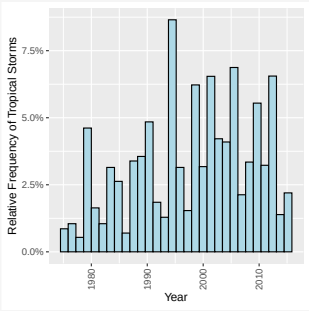

```
R> ggplot(data = storms, aes(x = year)) +
+   geom_histogram(aes(y=stat(count)/sum(..count..)),
+   fill = "lightblue", color = "black")+
+   scale_y_continuous(labels=scales::percent) +
+   ylab("Relative Frequency") + xlab("Year") +
+   theme(axis.text.x = element_text(angle = 90,
                                    vjust = 0.5, hjust=1))
```

Histogram: geom_histogram()

```
R> ggplot(data = storms, aes(x = year)) +
+   facet_wrap(~month)+
+   geom_histogram(fill = "lightblue",
+                  color = "black")+
+   ylab("Absolute Frequency") + xlab("Year") +
+   theme(axis.text.x = element_text(angle = 90,
                                    vjust = 0.5, hjust=1))
```

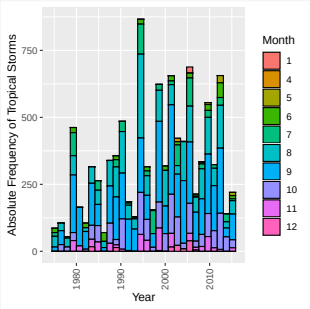



```
R> ggplot(data = storms, aes(x = year)) +
+   geom_histogram(aes(y=stat(count)/sum(..count..)),
+   fill = "lightblue", color = "black")+
+   scale_y_continuous(labels=scales::percent) +
+   ylab("Relative Frequency") + xlab("Year") +
+   theme(axis.text.x = element_text(angle = 90,
                                    vjust = 0.5, hjust=1))
```



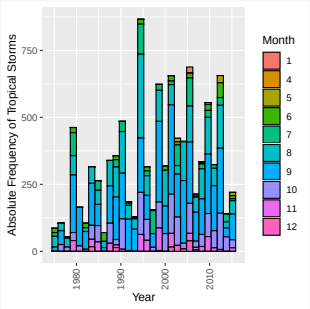
Histogram: geom_histogram()

```
R> ggplot(data = storms,
+        aes(x = year, fill = factor(month))) +
+   geom_histogram(color = "black",
+                  position = "stack")+
+   ylab("Absolute Frequency of Tropical Storms") +
+   xlab("Year") + labs(fill="Month") +
+   theme(axis.text.x = element_text(angle = 90,
+                         vjust = 0.5, hjust=1))
```



Histogram: geom_histogram()

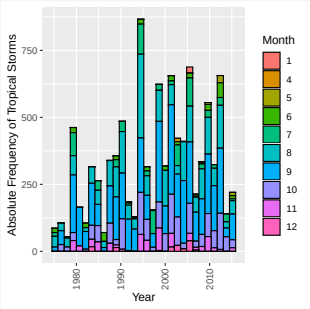
```
R> ggplot(data = storms,
+         aes(x = year, fill = factor(month))) +
+   geom_histogram(color = "black",
+                  position = "stack")+
+   ylab("Absolute Frequency of Tropical Storms") +
+   xlab("Year") + labs(fill="Month") +
+   theme(axis.text.x = element_text(angle = 90,
+                          vjust = 0.5, hjust=1))
```



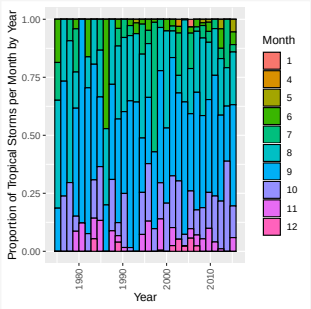

```
R> ggplot(data = storms,
+         aes(x = year, fill = factor(month))) +
+   geom_histogram(color = "black",
+                  position = "fill")+
+   ylab("Proportion of Tropical Storms per Month by Y
+   xlab("Year") + labs(fill="Month") +
+   theme(axis.text.x = element_text(angle = 90,
+                          vjust = 0.5, hjust=1))
```

Histogram: geom_histogram()

```
R> ggplot(data = storms,
+         aes(x = year, fill = factor(month))) +
+   geom_histogram(color = "black",
+                  position = "stack")+
+   ylab("Absolute Frequency of Tropical Storms") +
+   xlab("Year") + labs(fill="Month") +
+   theme(axis.text.x = element_text(angle = 90,
+                          vjust = 0.5, hjust=1))
```

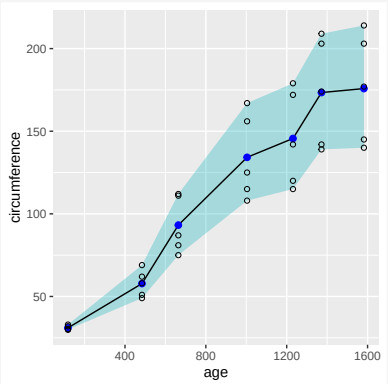


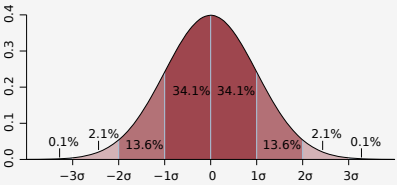

```
R> ggplot(data = storms,
+         aes(x = year, fill = factor(month))) +
+   geom_histogram(color = "black",
+                  position = "fill")+
+   ylab("Proportion of Tropical Storms per Month by Y
+   xlab("Year") + labs(fill="Month") +
+   theme(axis.text.x = element_text(angle = 90,
+                          vjust = 0.5, hjust=1))
```



stat_summary(): summary and ribbon plots

```
R> ggplot(data = Orange, aes(x = age, y = circumference)) +
+   stat_summary(fun = mean, geom = "ribbon", alpha = 0.3,
+                fun.max = max, fun.min = min, fill = "#00AFBB") +
+   stat_summary(fun = mean, geom ="line", colour = "black") +
+   stat_summary(fun = mean, geom ="point", colour = "blue", size = 2) +
+   geom_point(shape=1)
```

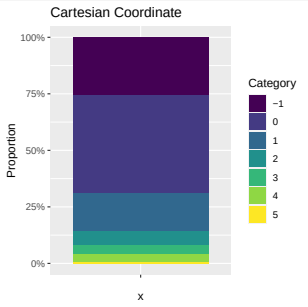


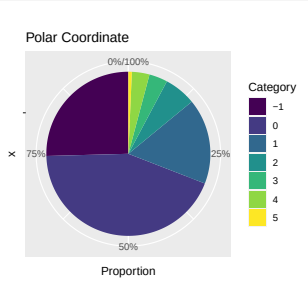


Coordinates

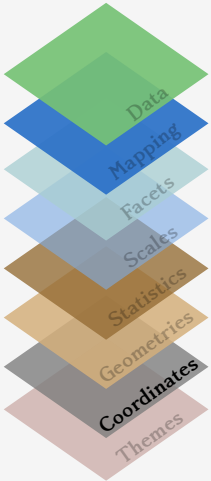
- Describes the space that data is projected onto
  - **cartesian** coordinates, **polar** coordinates, **map projections**, ...

```
R> p <- ggplot(data = storms, aes(x = "", fill = factor(category))) +
+       geom_bar(aes(y = stat(count)/sum(..count..))) +
+       scale_y_continuous(labels=scales::percent)+
+       labs(title = "Cartesian Coordinate", fill = "Category", y = "Proportion")
# Polar coordinate
R> p + coord_polar(theta="y") + labs(title = "Polar Coordinate")
```





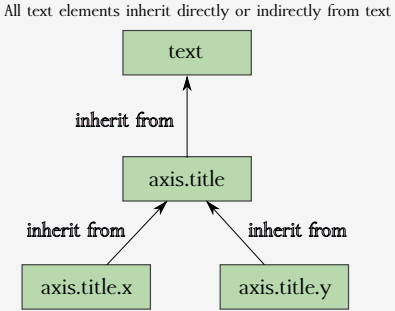
Decompose Graphics into its components



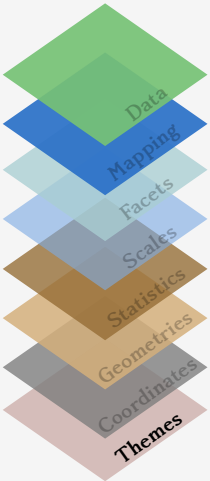
Themes

- Controls appearance of non-data elements (**helps make plot visually pleasing**)
  - titles, labels, fonts, background, gridlines, and legends

- Theme elements **inherit properties** from other theme elements hierarchically.

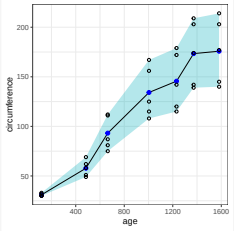


Decompose Graphics into its components

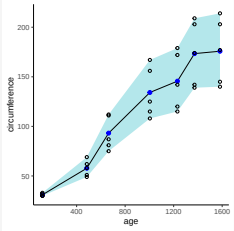


Standard Themes

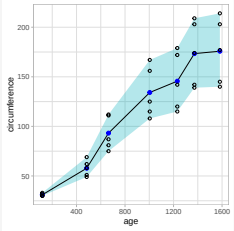
```
1 R> p + theme_bw()
```



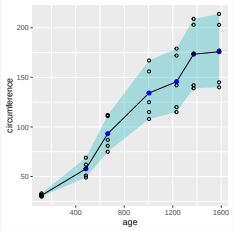

```
1 R> p + theme_classic()
```



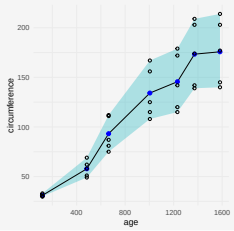

```
1 R> p + theme_light()
```



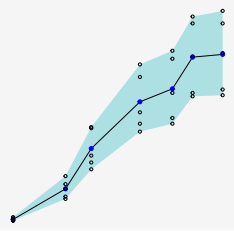

```
1 R> p + theme_grey()
```




```
1 R> p + theme_minimal()
```

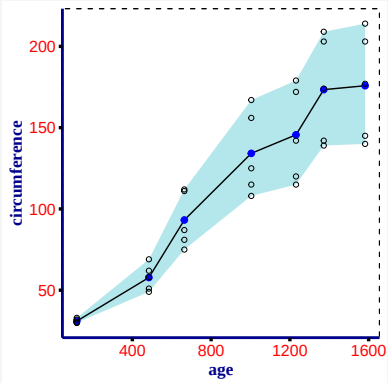



```
1 R> p + theme_void()
```



Theme: axis and text modification

```
R> p +
+    theme(panel.grid.major = element_blank(),
+          panel.grid.minor = element_blank(),
+          panel.border = element_rect(fill=NA, color="black",
+                                      size=0.5, linetype="dashed"),
+          axis.line = element_line(colour = "darkblue",
+                                   size = 1, linetype = "solid"),
+          axis.ticks = element_line(color="black", size=1.2),
+          panel.background = element_rect(fill = "white"),
+          axis.title = element_text(size = 13, colour = "darkblue",
+                                    family="serif", face = "bold"),
+          axis.text = element_text(size=12, colour = "red"))
```

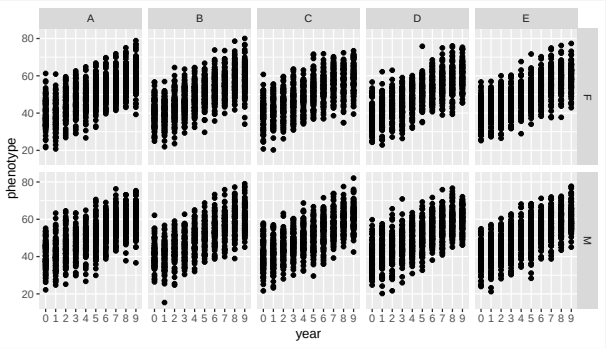


qplot()- Quick Plot

- qplot() developed as a convenient wrapper for creating **basic plots**
- ggplot() developed to create **basic and complex plots**

```
R> qplot(year, phenotype, data = cbp,
         facets = sex ~ herd)
```

```
R> p + ggplot(data = cbp, aes(x = year, y = phenotype)) +
+    geom_point() +
+    facet_grid(rows = vars(sex), cols = vars(herd))
```



Annotation

Annotation

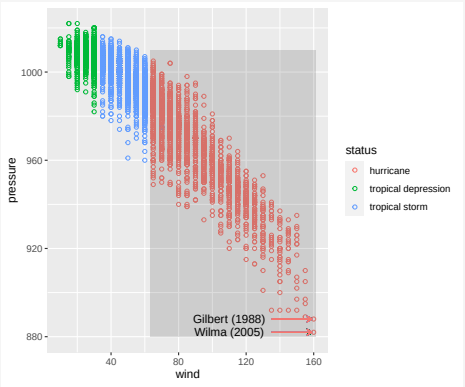
- Add a **new layer** to the plot **controlled by `geom` argument**
- Annotate layers does not map from variables of a data frame
  - **create new vectors which is passed to the plot**
- Layers created with this function will **never affect the legend**

```
annotate(
  geom,
  x = NULL,
  y = NULL,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
  ymax = NULL,
  xend = NULL,
  yend = NULL,
  ...,
  na.rm = FALSE
)
```

geom	name of geom to use for annotation
x, y, xmin, ymin, xmax, ymax, xend, yend	positioning aesthetics - you must specify at least one of these.
...	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.
na.rm	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

Annotation: examples

```
R> ggplot(storms, aes(x = wind, y = pressure, colour = status)) +
+   geom_point(shape=1) +
+   annotate("rect", xmin = 63, xmax = 161, ymin = 880,
+                              ymax = 1010, alpha = .2) +
+   annotate("text", x = 110, y = 888, label = "Gilbert (1988)") +
+   annotate("text", x = 110, y = 882, label = "Wilma (2005)") +
+   geom_segment(aes(x = 135, y = 888, xend = 159, yend = 888),
+                arrow = arrow(length = unit(0.2, "cm")),
+                show.legend = FALSE)+
+   geom_segment(aes(x = 135, y = 882, xend = 159, yend = 882),
+                arrow = arrow(length = unit(0.2, "cm")),
+                show.legend = FALSE)
```



Annotation: linear regression (ggpmisc and egg packages)

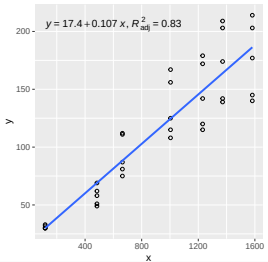
```
# Function to extract coefficients lm(y~x)
R> colnames(Orange)[2:3] <- c("x","y")
R> p1 <-
+   ggplot(data = Orange, aes(x = x, y = y)) +
+    geom_point(shape=1) +
+    geom_smooth(method = "lm", se = FALSE) +
+    ggpmisc::stat_poly_eq(
+     aes(label =  paste(after_stat(eq.label),
+     after_stat(adj.rr.label),
+     sep = "*\", \"*")),
+     formula = y ~ poly(x, 1, raw = TRUE))
```

- egg::tag_facet(): adds a dummy text layer to a ggplot to label facets and sets facet strips to blank.

```
R> p2 <- p1 + facet_wrap(~Tree)
R> tag_facet(p2, x = 1300, y = -Inf, vjust = -1,
+   open = "", close = ")", tag_pool = LETTERS)
```

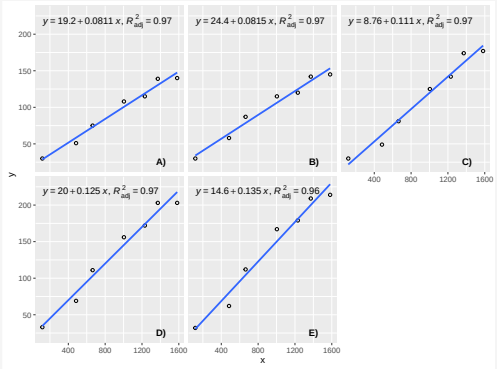
Annotation: linear regression (ggpmisc and egg packages)

```
# Function to extract coefficients lm(y~x)
R> colnames(Orange)[2:3] <- c("x","y")
R> p1 <-
+   ggplot(data = Orange, aes(x = x, y = y)) +
+    geom_point(shape=1) +
+    geom_smooth(method = "lm", se = FALSE) +
+    ggpmisc::stat_poly_eq(
+     aes(label =  paste(after_stat(eq.label),
+     after_stat(adj.rr.label),
+     sep = "*\", \"*")),
+     formula = y ~ poly(x, 1, raw = TRUE))
```



- egg::tag_facet(): adds a dummy text layer to a ggplot to label facets and sets facet strips to blank.

```
R> p2 <- p1 + facet_wrap(~Tree)
R> tag_facet(p2, x = 1300, y = -Inf, vjust = -1,
+   open = "", close = ")", tag_pool = LETTERS)
```



geom_text() and geom_label()

- geom_text() adds only text to the plot.
- geom_label() draws a rectangle behind the text, making it easier to read.

```
R> ex1 <- data.frame(y = 1:3, family = c("sans", "serif", "Roboto"))
R> gplot(data = ex1) +
+   geom_text(aes(x=1, y=y, label = family,
+             colour = family, family = family),
+             show.legend = FALSE, size = 6)+
+   geom_label(aes(x=0, y=y, label = family,
+              family = family, fontface = c(1:3)))+
+   xlim(c(-0.5,1.5)) +
+   theme_classic(base_size = 12)
```

geom_text() and geom_label()

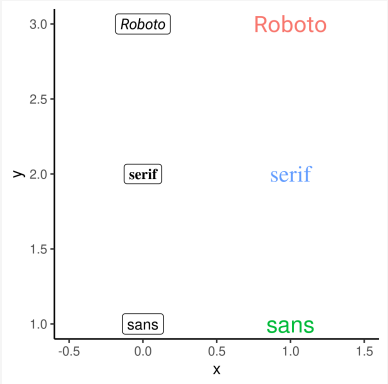
- geom_text() adds only text to the plot.
- geom_label() draws a rectangle behind the text, making it easier to read.

```
R> ex1 <- data.frame(y = 1:3, family = c("sans", "serif", "Roboto"))
R> gplot(data = ex1) +
+   geom_text(aes(x=1, y=y, label = family,
+             colour = family, family = family),
+             show.legend = FALSE, size = 6)+
+   geom_label(aes(x=0, y=y, label = family,
+              family = family, fontface = c(1:3)))+
+   xlim(c(-0.5,1.5)) +
+   theme_classic(base_size = 12)
```

geom_text() and geom_label()

- geom_text() adds only text to the plot.
- geom_label() draws a rectangle behind the text, making it easier to read.

```
R> ex1 <- data.frame(y = 1:3, family = c("sans", "serif", "Roboto"))
R> gplot(data = ex1) +
+   geom_text(aes(x=1, y=y, label = family,
+             colour = family, family = family),
+             show.legend = FALSE, size = 6)+
+   geom_label(aes(x=0, y=y, label = family,
+              family = family, fontface = c(1:3)))+
+   xlim(c(-0.5,1.5)) +
+   theme_classic(base_size = 12)
```



Plot Composition

Plot Composition: `pathwork`

- **Combine separate ggplots** into the same graphic.
- Let we assume `p1`, `p2`, and `p3` are ggplot graphics.

```
R> p1|(p3/p2)
```

```
R> (p3 | p2) + plot_annotation(tag_levels = "A")
```

Plot Composition: pathwork

- **Combine separate ggplots** into the same graphic.
- Let we assume p1, p2, and p3 are ggplot graphics.

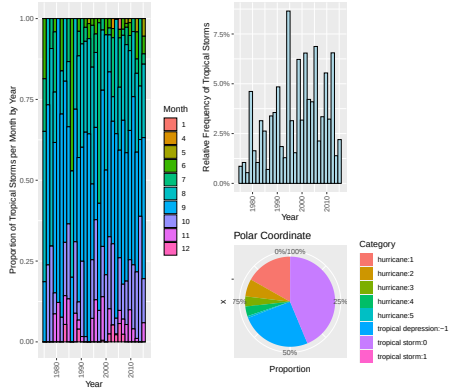
```
R> p1|(p3/p2)
```

```
R> (p3 | p2) + plot_annotation(tag_levels = "A")
```

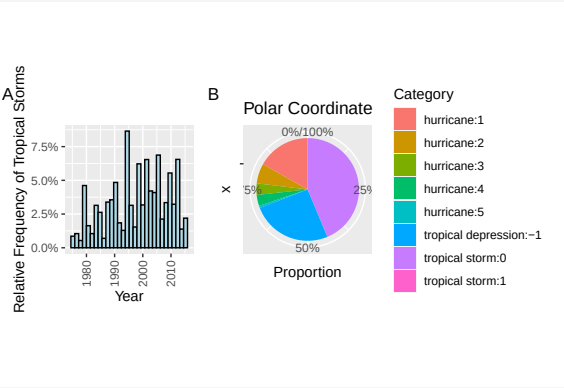
Plot Composition: pathwork

- **Combine separate ggplots** into the same graphic.
- Let we assume p1, p2, and p3 are ggplot graphics.

```
1 R> p1|(p3/p2)
```




```
1 R> (p3 | p2) + plot_annotation(tag_levels = "A")
```



Plot Composition: `pathwork`, `gridExtra`, and `ggpubr`

```
R> p1 | gridExtra::tableGrob(OrangeSummary)
```

```
R> text <- paste("The Orange data frame has 35 rows and
+                3 columns", "of records of the growth
+                of orange trees.", sep = "\n")
R> p4 <- wrap_elements(
+    ggpubr::text_grob(text, face="bold", color = "blue"))
R> p4/( p1 | gridExtra::tableGrob(OrangeSummary))
```

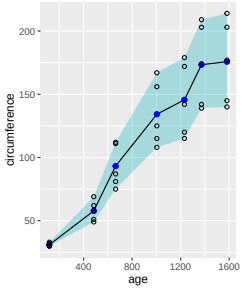
Plot Composition: pathwork, gridExtra, and ggpubr

```
R> p1 | gridExtra::tableGrob(OrangeSummary)
```

```
R> text <- paste("The Orange data frame has 35 rows and
+                3 columns", "of records of the growth
+                of orange trees.", sep = "\n")
R> p4 <- wrap_elements(
+    ggpubr::text_grob(text, face="bold", color = "blue"))
R> p4/( p1 | gridExtra::tableGrob(OrangeSummary))
```

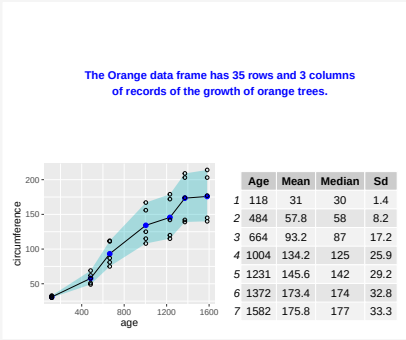
Plot Composition: pathwork, gridExtra, and ggpubr

```
R> p1 | gridExtra::tableGrob(OrangeSummary)
```



	Age	Mean	Median	Sd
1	118	31	30	1.4
2	484	57.8	58	8.2
3	664	93.2	87	17.2
4	1004	134.2	125	25.9
5	1231	145.6	142	29.2
6	1372	173.4	174	32.8
7	1582	175.8	177	33.3

```
R> text <- paste("The Orange data frame has 35 rows and
+               3 columns", "of records of the growth
+               of orange trees.", sep = "\n")
R> p4 <- wrap_elements(
+    ggpubr::text_grob(text, face="bold", color = "blue"))
R> p4/( p1 | gridExtra::tableGrob(OrangeSummary))
```



	Age	Mean	Median	Sd
1	118	31	30	1.4
2	484	57.8	58	8.2
3	664	93.2	87	17.2
4	1004	134.2	125	25.9
5	1231	145.6	142	29.2
6	1372	173.4	174	32.8
7	1582	175.8	177	33.3

References

- Wickham H. A layered grammar of graphics. **Journal of Computational and Graphical Statistics**, v. 19, no. 1, p. 3-28, 2010
- Wickham H. **ggplot2: Elegant Graphics for Data Analysis**. Springer, 2016.

Useful links:

- Scales and guides: `https://ggplot2tor.com/scales/`
- `scale_colour_*`: Viridis colour; Sci-Fi Themed; colorblind
- The R graph Galery: `https://www.r-graph-gallery.com/ggplot2-package.html`
- RDocumentation: `https://www.rdocumentation.org/packages/ggplot2/versions/3.3.5`