Big Data Seminar

RStudio & ggplot2

Members

- Đoàn Ngọc Mai - 21127104
- Lê Nguyễn Kiều Oanh - 21127129
- Dương Trường Bình - 21127229
- Lê Phước Quang Huy - 21127616

Lecturers

- Đỗ Trọng Lễ
- Bùi Huỳnh Trung Nam
- Lê Ngọc Thành
- Nguyễn Ngọc Thảo

Ho Chi Minh City, August 2024

CONTENT

01 INTRODUCTION

02 GRAMMAR

03 LAYERS

04 SCALES

Statistical computing and graphics

R Foundation for Statistical Computing

Main Function

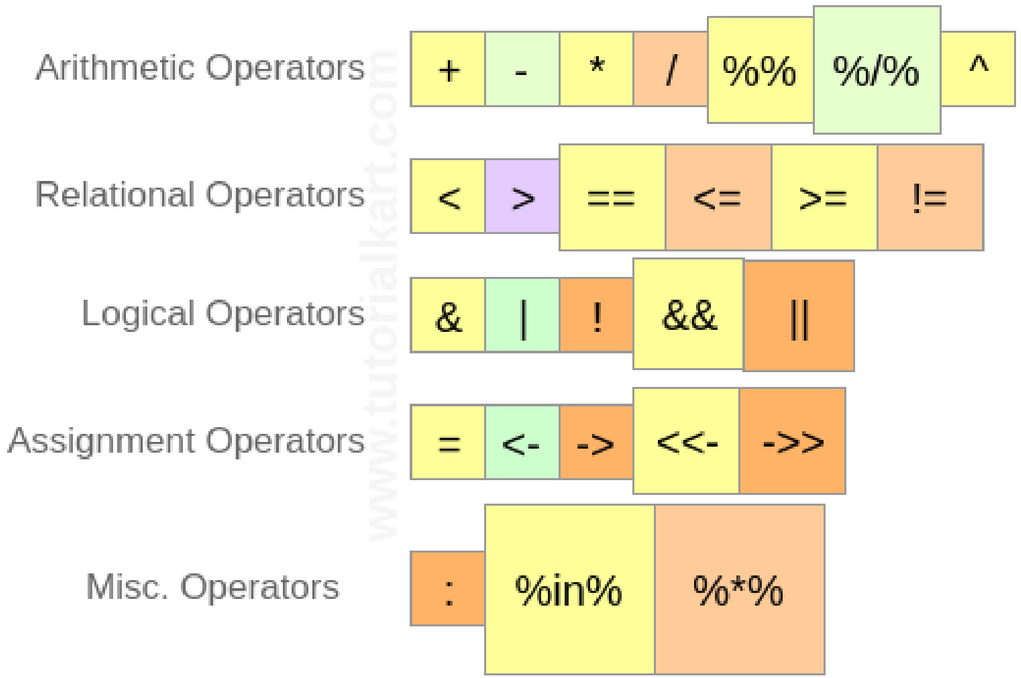
- Statistical Analysis
- Data Visualization
- Data Processing
- Machine Learning

Advantages

- Free and Open Source
- Large Community
- Variety of Packages
- High Quality Graphs

Ross Ihaka

Robert Gentleman



https://www.r-project.org/

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

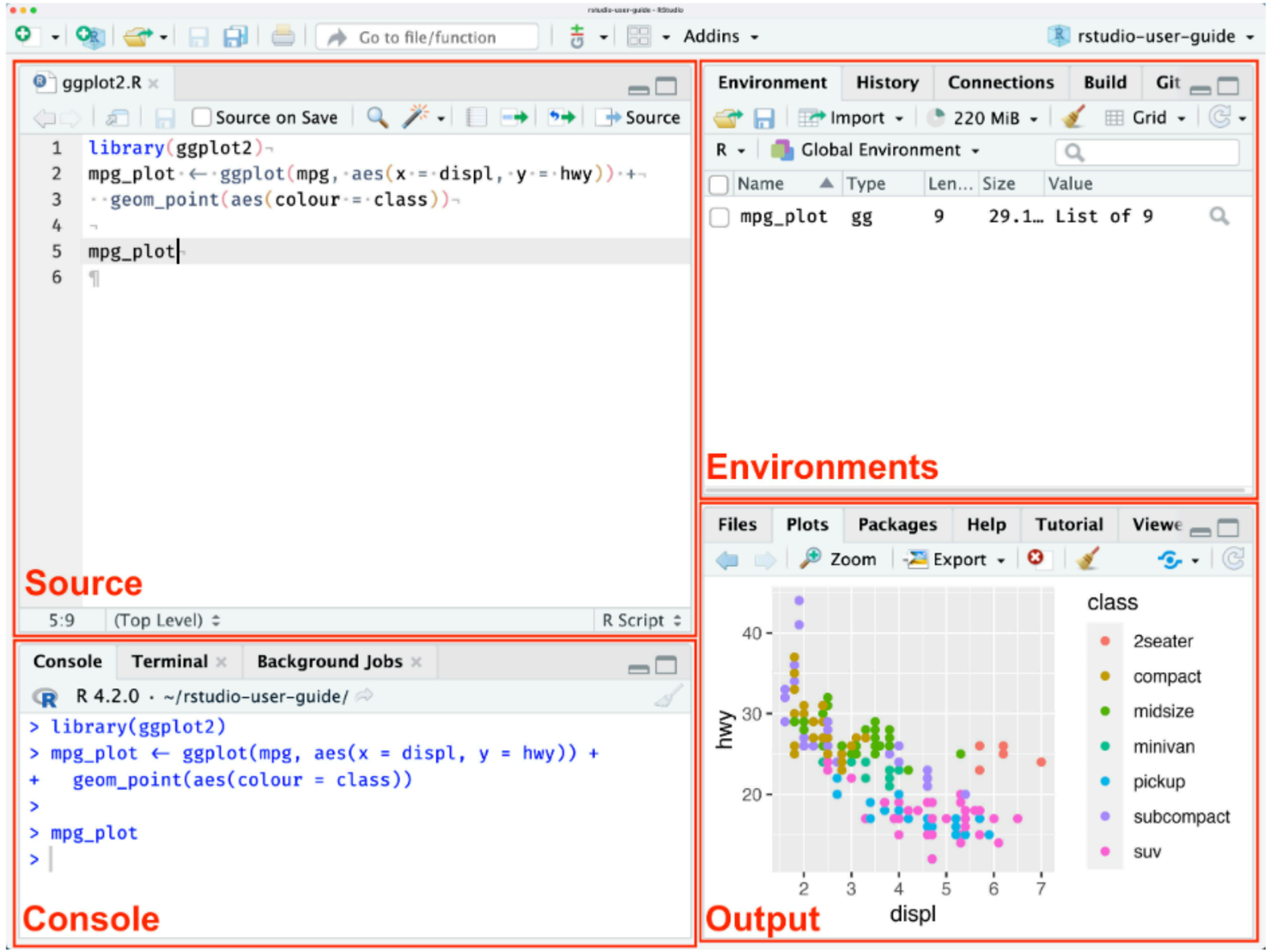
Functions

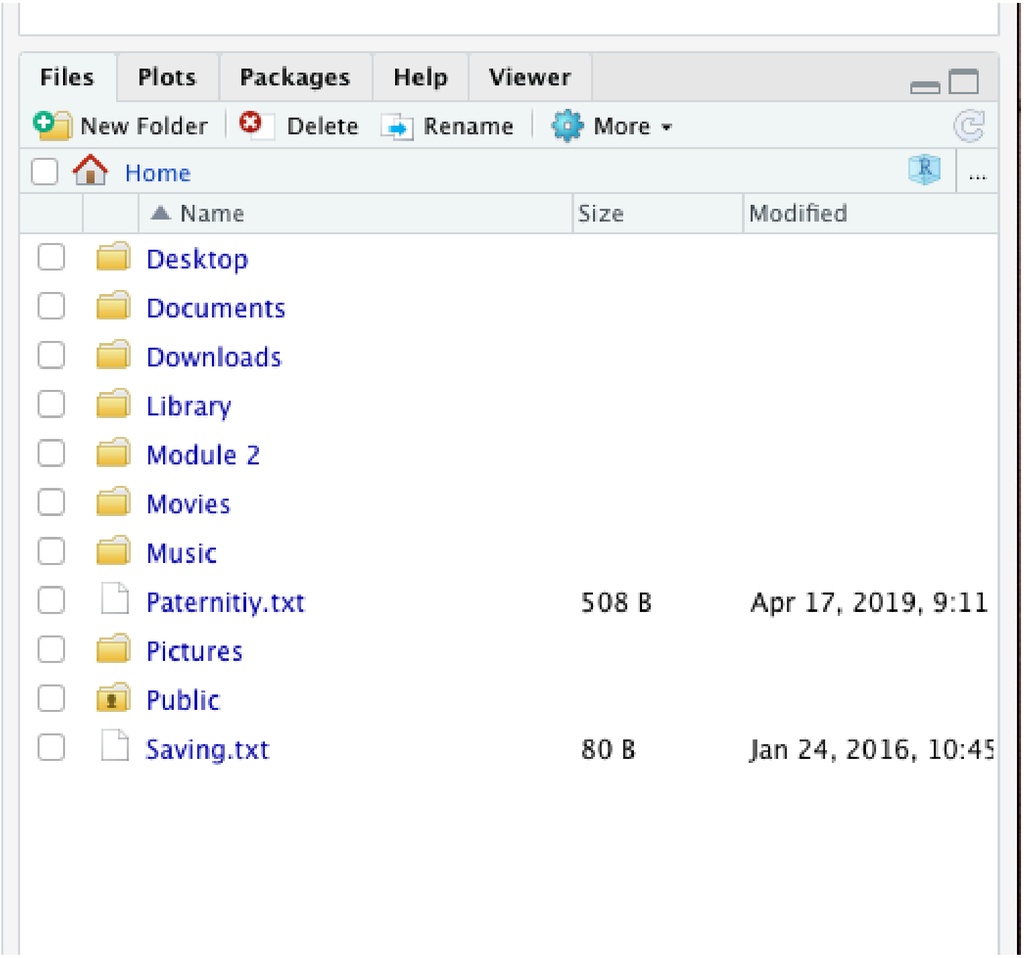
```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

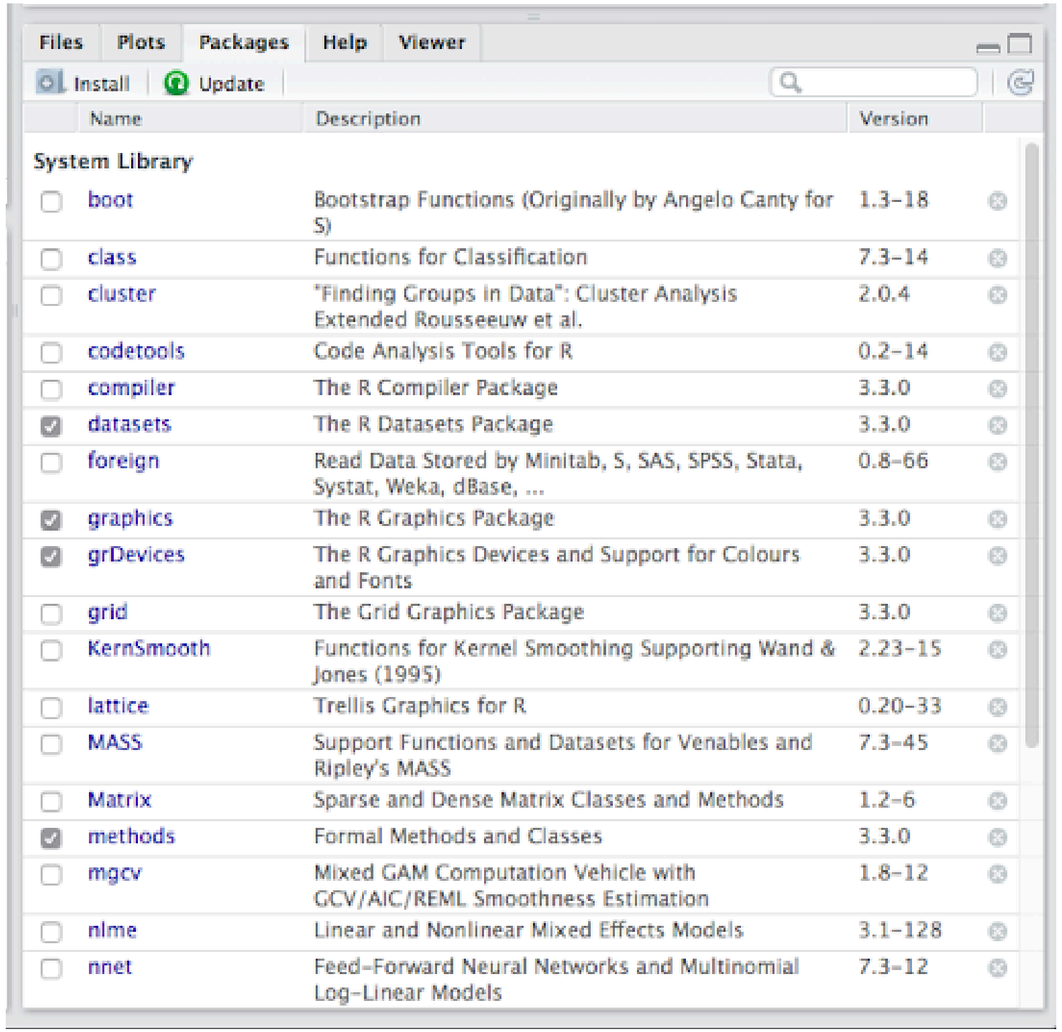
RStudio


Go to file/function
Addins
rstudio-user-guide
ggplot2.R
Source on Save
Source
library(ggplot2)
mpg_plot ← ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(colour = class))
mpg_plot
Source
5:9
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.2.0 · ~/rstudio-user-guide/
> library(ggplot2)
> mpg_plot ← ggplot(mpg, aes(x = displ, y = hwy)) +
+   geom_point(aes(colour = class))
>
> mpg_plot
>
Console
Environment
History
Connections
Build
Import
220 MiB
Grid
R
Global Environment
Name
Type
Len...
Size
Value
mpg_plot
gg
9
29.1...
List of 9
Environments
Files
Plots
Packages
Help
Tutorial
Zoom
Export
hwy
displ
class
2seater
compact
midsize
minivan
pickup
subcompact
suv
Output


Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
Desktop
Documents
Downloads
Library
Module 2
Movies
Music
Paternitiy.txt
508 B
Apr 17, 2019, 9:11
Pictures
Public
Saving.txt
80 B
Jan 24, 2016, 10:45

Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
10
5
number

RStudio


Files
Plots
Packages
Help
Viewer
Install
Update
Name
Description
Version
System Library
boot
Bootstrap Functions (Originally by Angelo Canty for S)
1.3-18
class
Functions for Classification
7.3-14
cluster
"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.
2.0.4
codetools
Code Analysis Tools for R
0.2-14
compiler
The R Compiler Package
3.3.0
datasets
The R Datasets Package
3.3.0
foreign
Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...
0.8-66
graphics
The R Graphics Package
3.3.0
grDevices
The R Graphics Devices and Support for Colours and Fonts
3.3.0
grid
The Grid Graphics Package
3.3.0
KernSmooth
Functions for Kernel Smoothing Supporting Wand & Jones (1995)
2.23-15
lattice
Trellis Graphics for R
0.20-33
MASS
Support Functions and Datasets for Venables and Ripley's MASS
7.3-45
Matrix
Sparse and Dense Matrix Classes and Methods
1.2-6
methods
Formal Methods and Classes
3.3.0
mgcv
Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation
1.8-12
nlme
Linear and Nonlinear Mixed Effects Models
3.1-128
nnet
Feed-Forward Neural Networks and Multinomial Log-Linear Models
7.3-12

DATASET

House Dataset

Contains 1,134 records of apartments for sale in Ho Chi Minh City from 'chotot', with 15 columns of information.

DiaChi	GiaBan	DuAn	DienTich	TinhTrangBDS	Gia/m2	PhongNgu	PhongVeSinh	SoTang	TinhTrangGiayTo	TinhTrangNoiThat	HuongBanCong	HuongCua	DacDiem	Quan

IMDB Indian Movies Dataset

A comprehensive collection of 15,506 Indian films with 10 attributes including titles, release years, genres, directors, actors, and IMDb ratings.

Name	Year	Duration	Genre	Rating	Votes	Director	Actor 1	Actor 2	Actor 3

GRAMMAR

01 DATA

02 AESTHETIC

03 GEOMETRY

04 ADDING COLOUR

05 SCALE

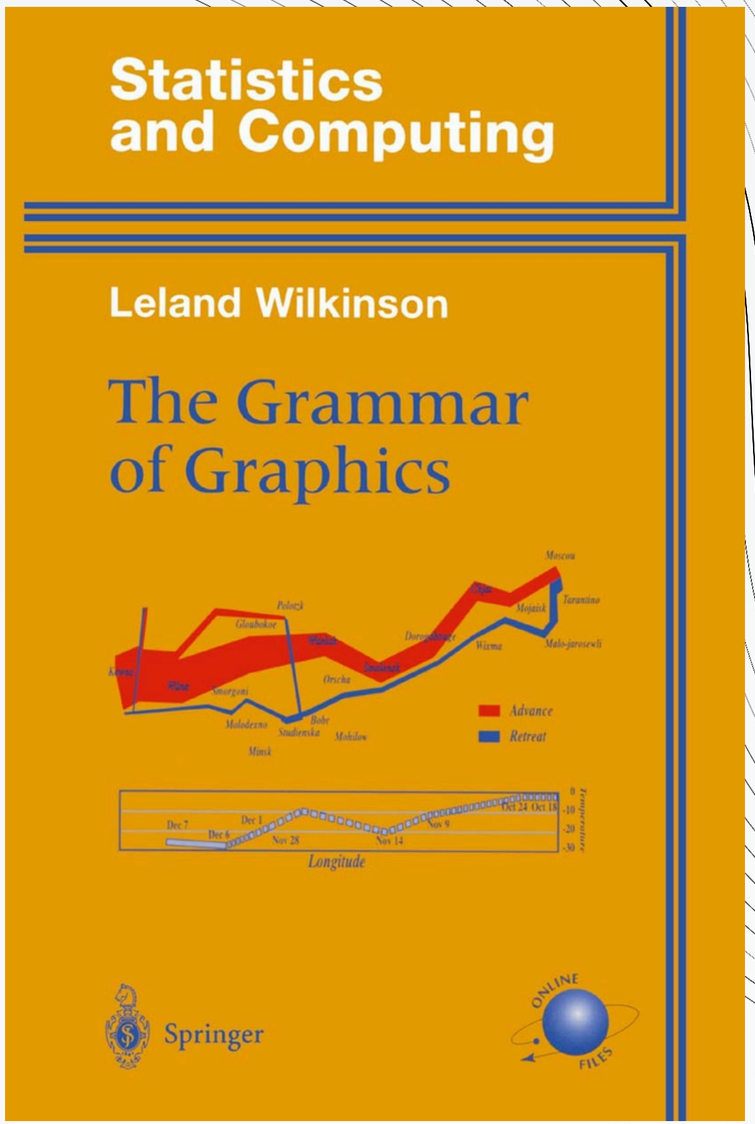
06 FACET

07 THEME

GRAMMAR

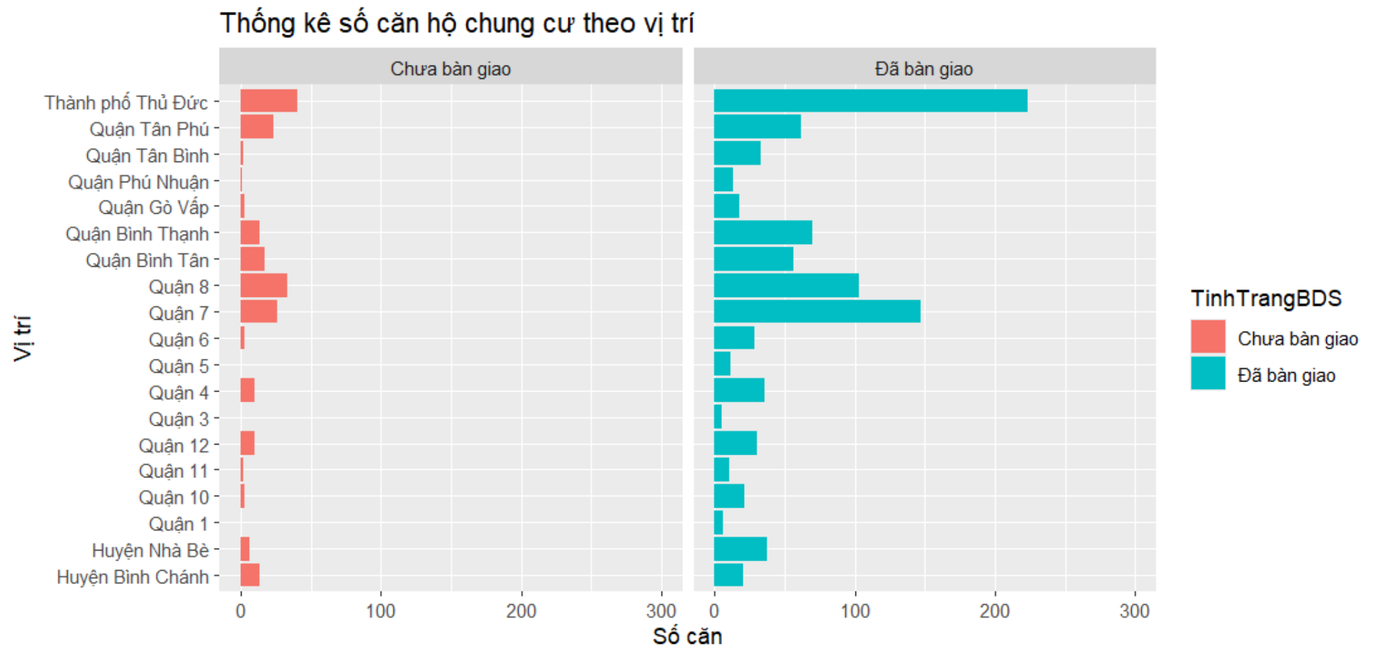
- The "gg" in "ggplot2" comes from "The Grammar of Graphics," a book published in 1999 that formalized a graphical language system to represent charts consistently.

- The "ggplot2" package appeared in R around 2007



GRAMMAR

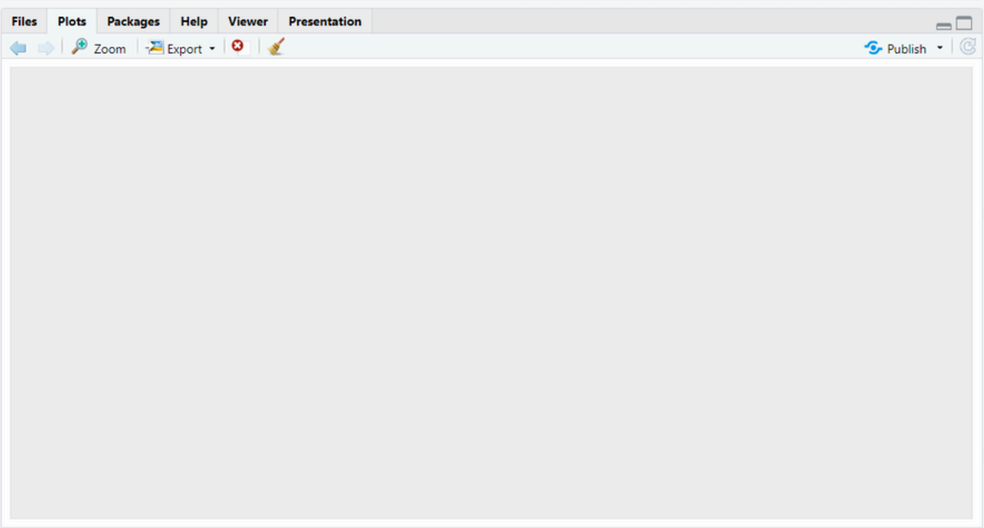
DATA	Data used
AESTHETIC	Which variables go where
GEOMETRY	Type of plot
FACET	Different panels for different things
SCALE	How far the axes extend
THEME	What font and color are used in labeling



GRAMMAR | Data

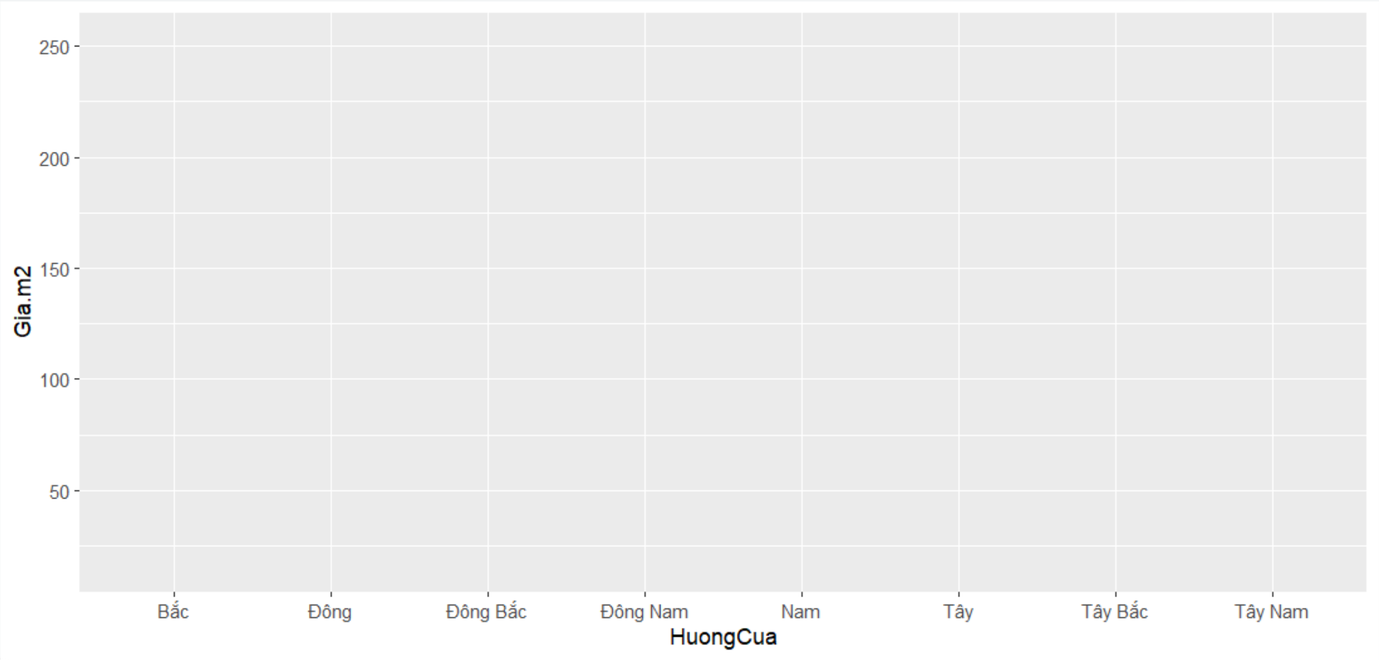
```
ggplot(data = house_df)
```

DuAn	DienTich	TinhTrangBDS	Gia.m2	PhongNgu	PhongVeSi
The Art	66.00	Chưa bàn giao	35.61	2	
Celadon City	71.00	Đã bàn giao	48.17	2	
Akari City	56.00	Đã bàn giao	42.86	2	
One Verandah Mapletree	107.00	Đã bàn giao	70.09	3	
Akari City	75.00	Chưa bàn giao	41.33	2	
Celadon City	104.00	Đã bàn giao	54.81	3	
Căn hộ D'Lusso	60.00	Đã bàn giao	50.00	2	
Orchid Park	72.00	Đã bàn giao	24.86	2	
Akari City	97.00	Đã bàn giao	42.27	3	
La Astoria	60.00	Đã bàn giao	38.33	2	
Võ Đình Apartment	90.00	Đã bàn giao	24.78	2	
New Saigon-Hoàng Anh Gia Lai 3	100.00	Đã bàn giao	22.50	2	
Shizen Home (Asiana Riverside)	77.50	Chưa bàn giao	64.52	2	



GRAMMAR | Aesthetic

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2))
```

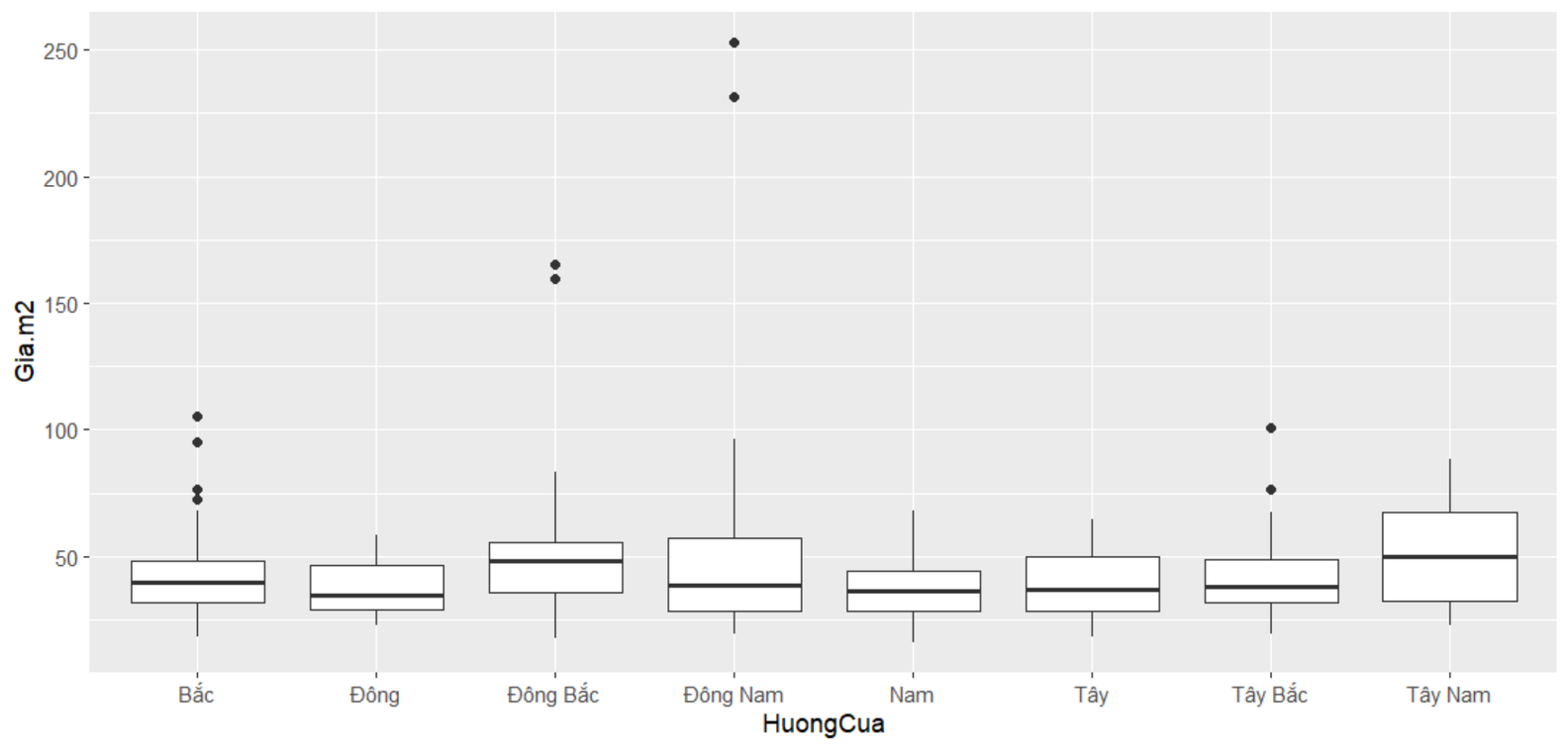


- Assign variables to the x-axis and y-axis

GRAMMAR | Geometry

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_boxplot()
```

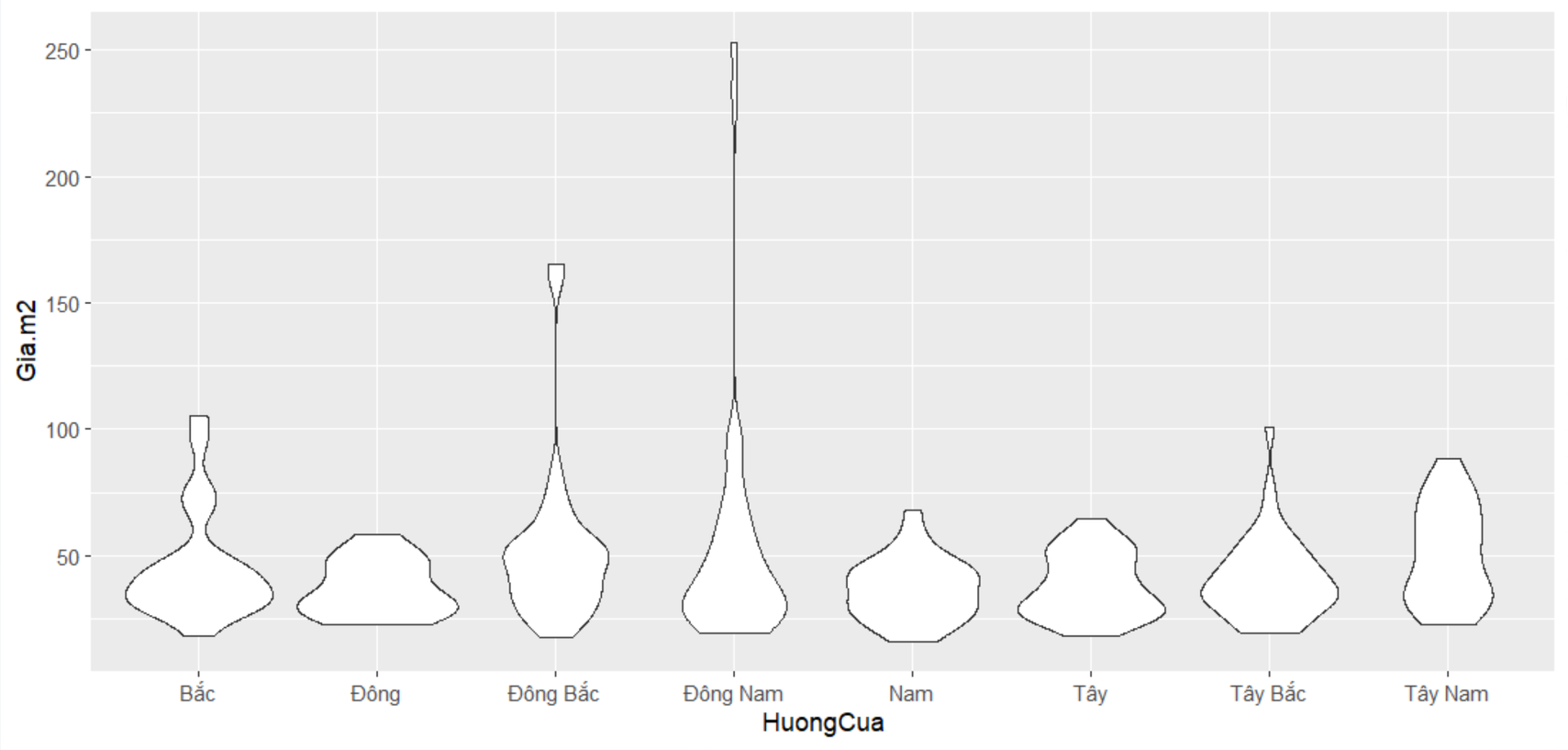
- Adding Geom layer to Base layer



GRAMMAR | Geometry

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_violin()
```

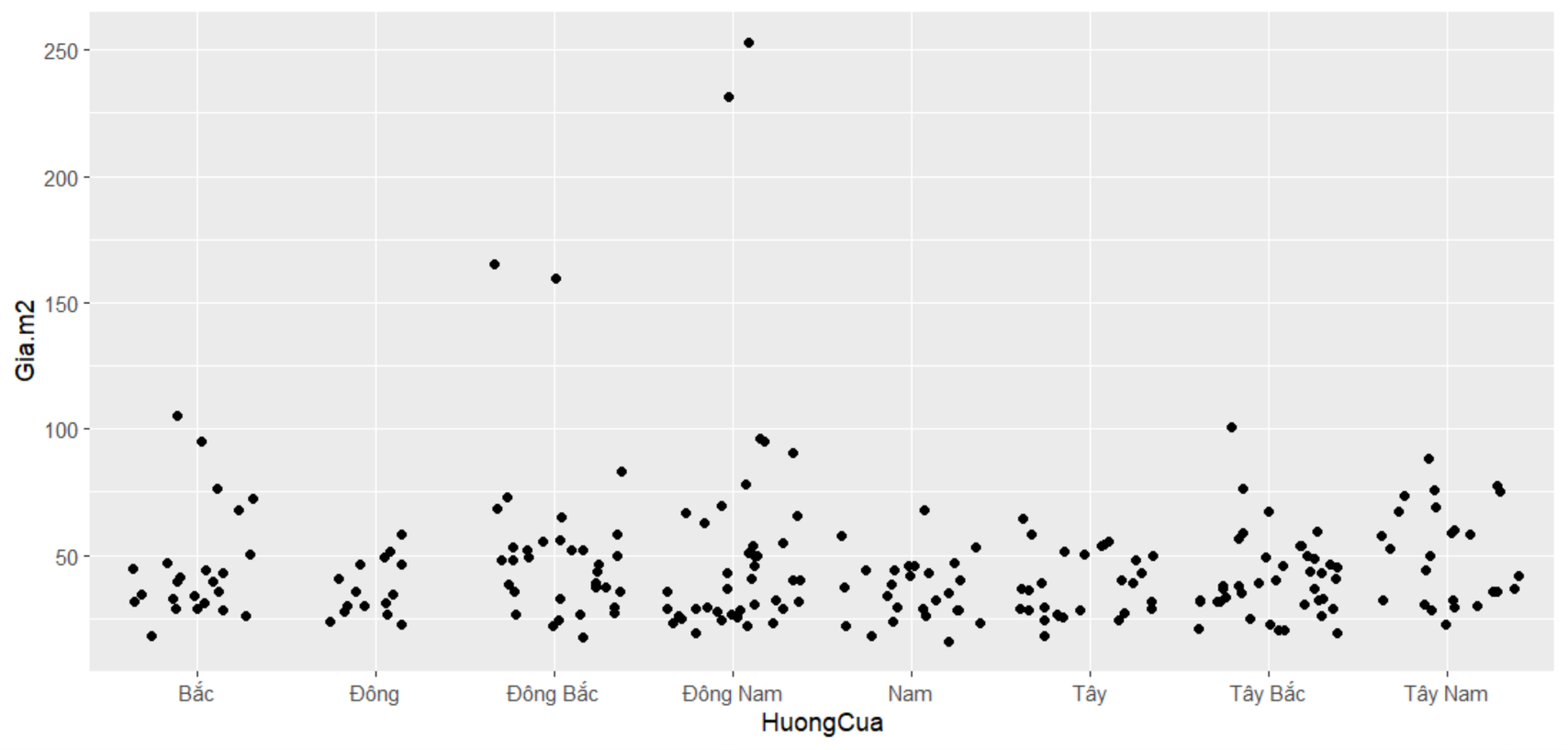
- Adding Geom layer to Base layer



GRAMMAR | Geometry

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_jitter()
```

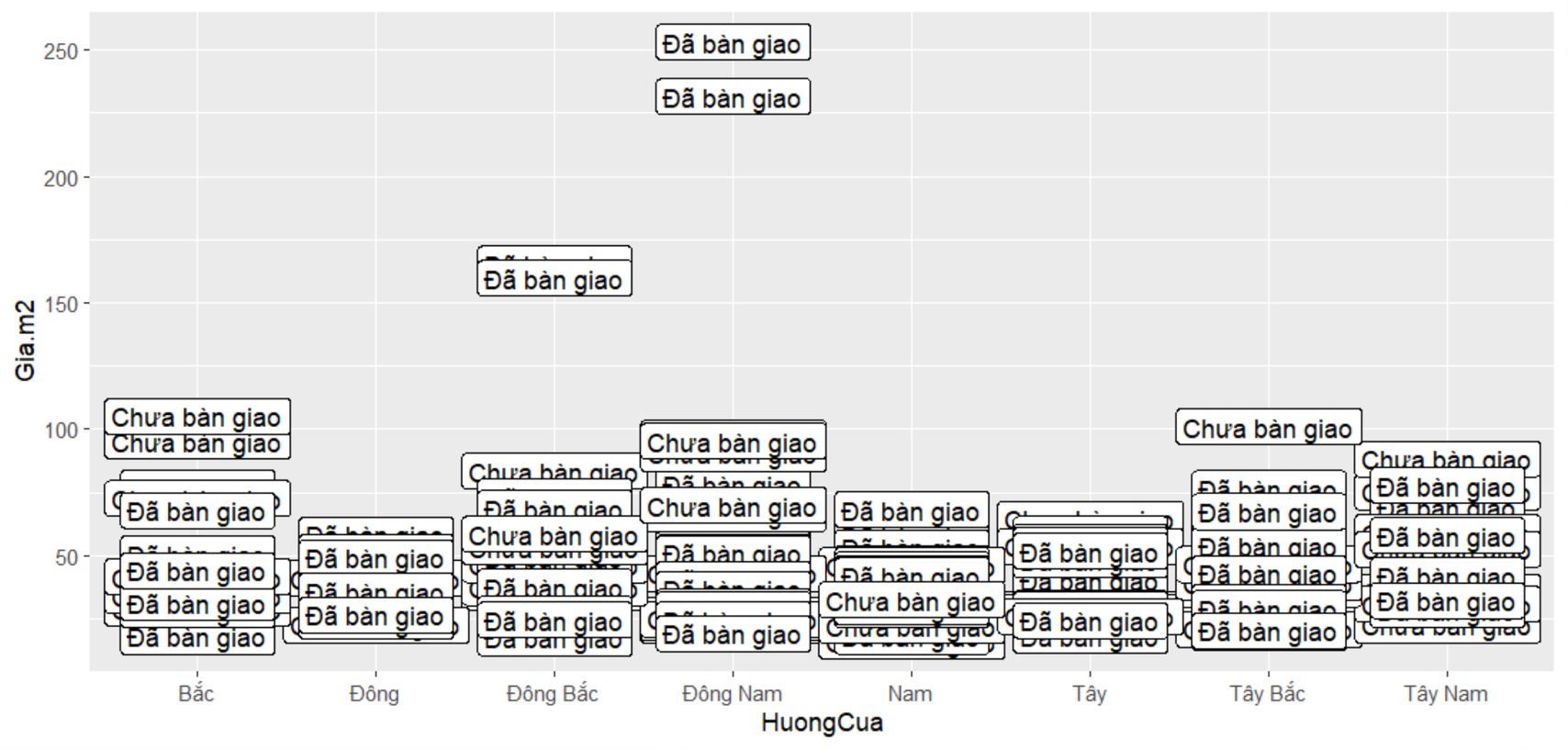
- Adding Geom layer to Base layer



GRAMMAR | Geometry

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_label()
```

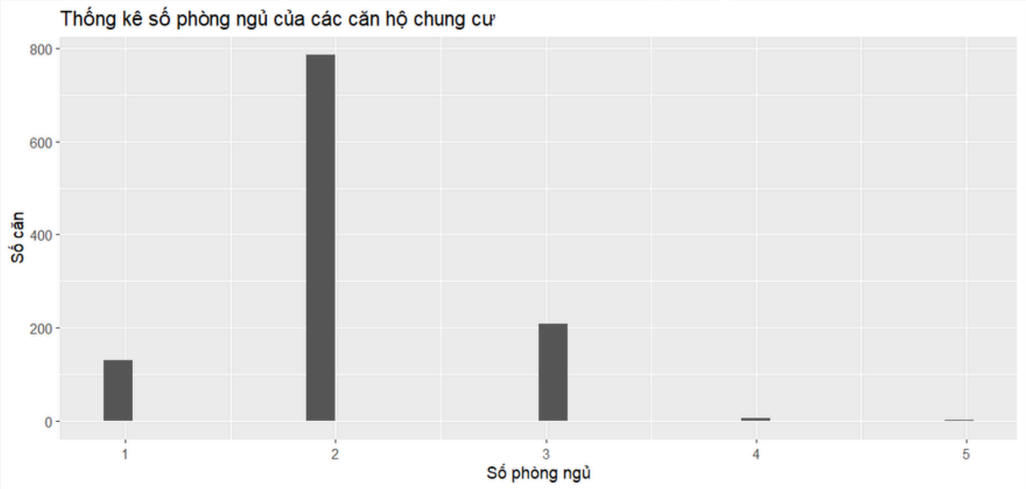
- Adding Geom layer to Base layer



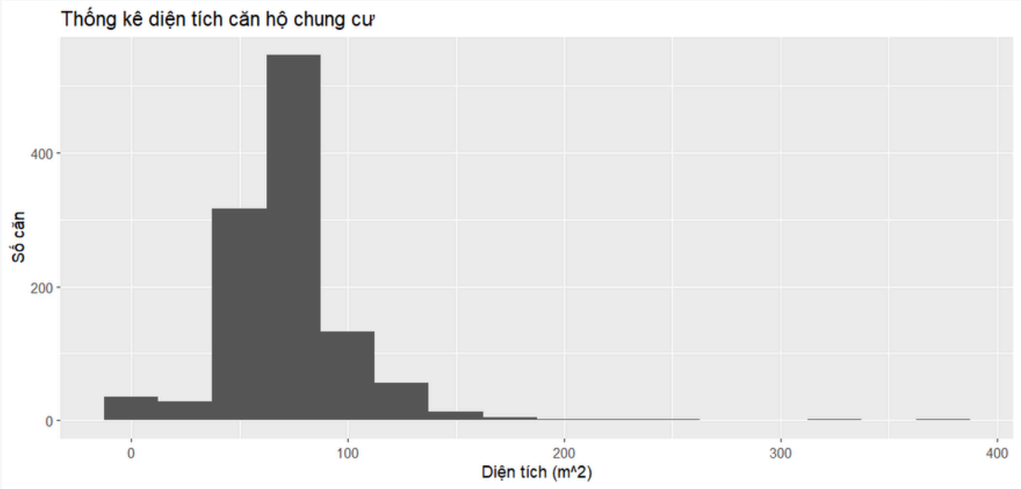
GRAMMAR | Geometry

- There are geom functions that support the presentation of 1-variable charts such as:

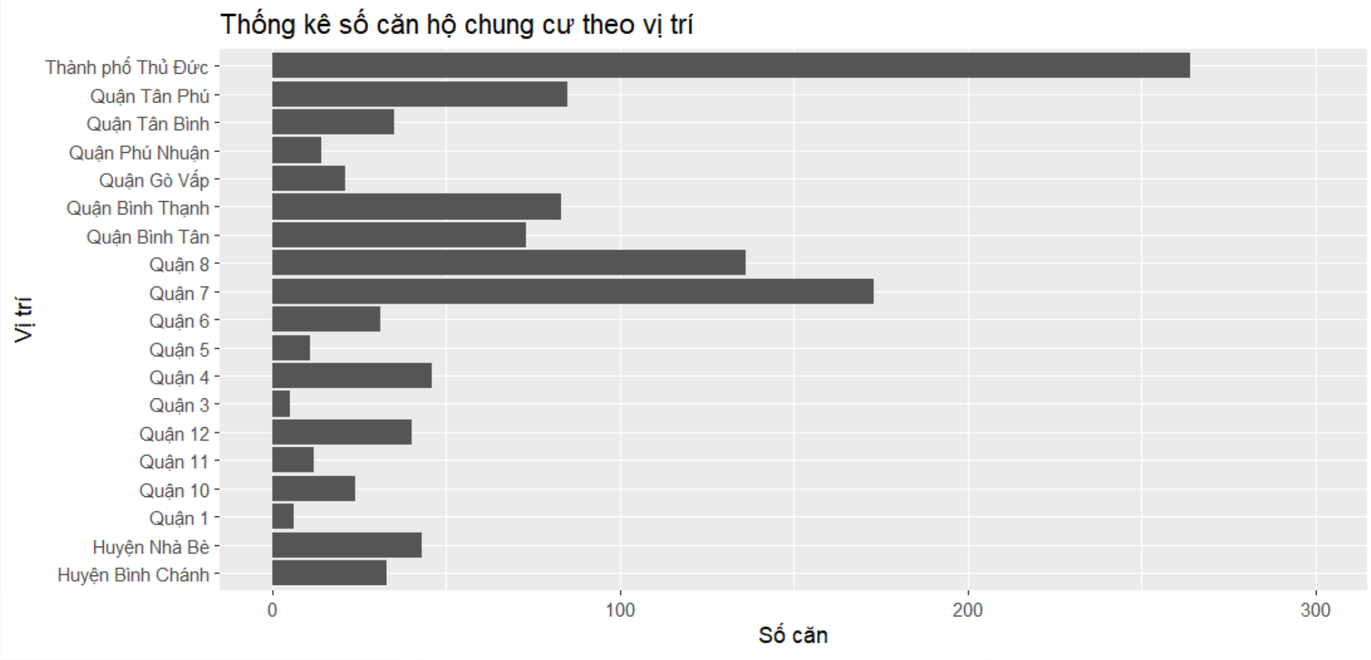
base_plot + geom_histogram()



base_plot + geom_histogram(binwidth=25)



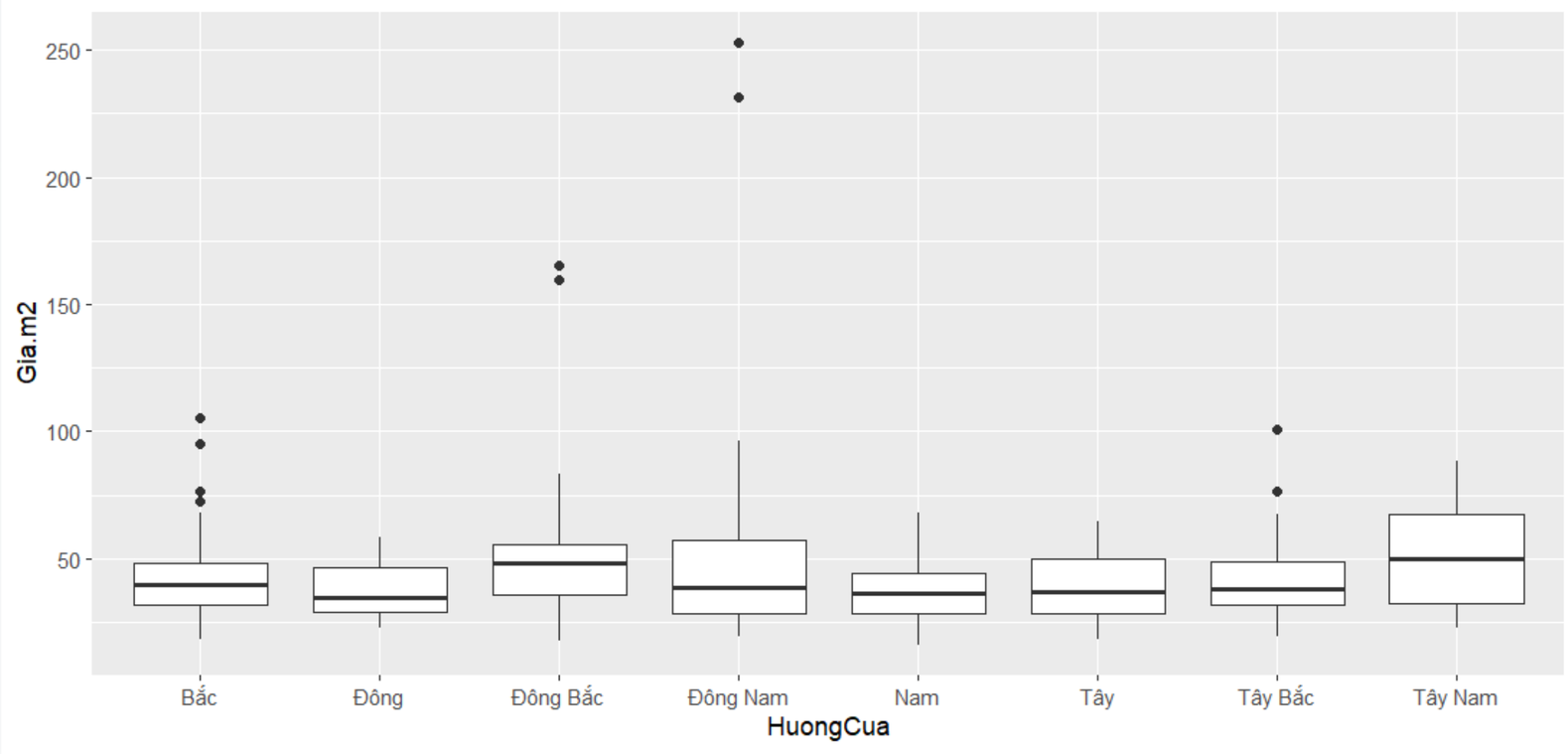
base_plot + geom_bar() + coord_flip()



GRAMMAR | Adding color

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_boxplot()
```

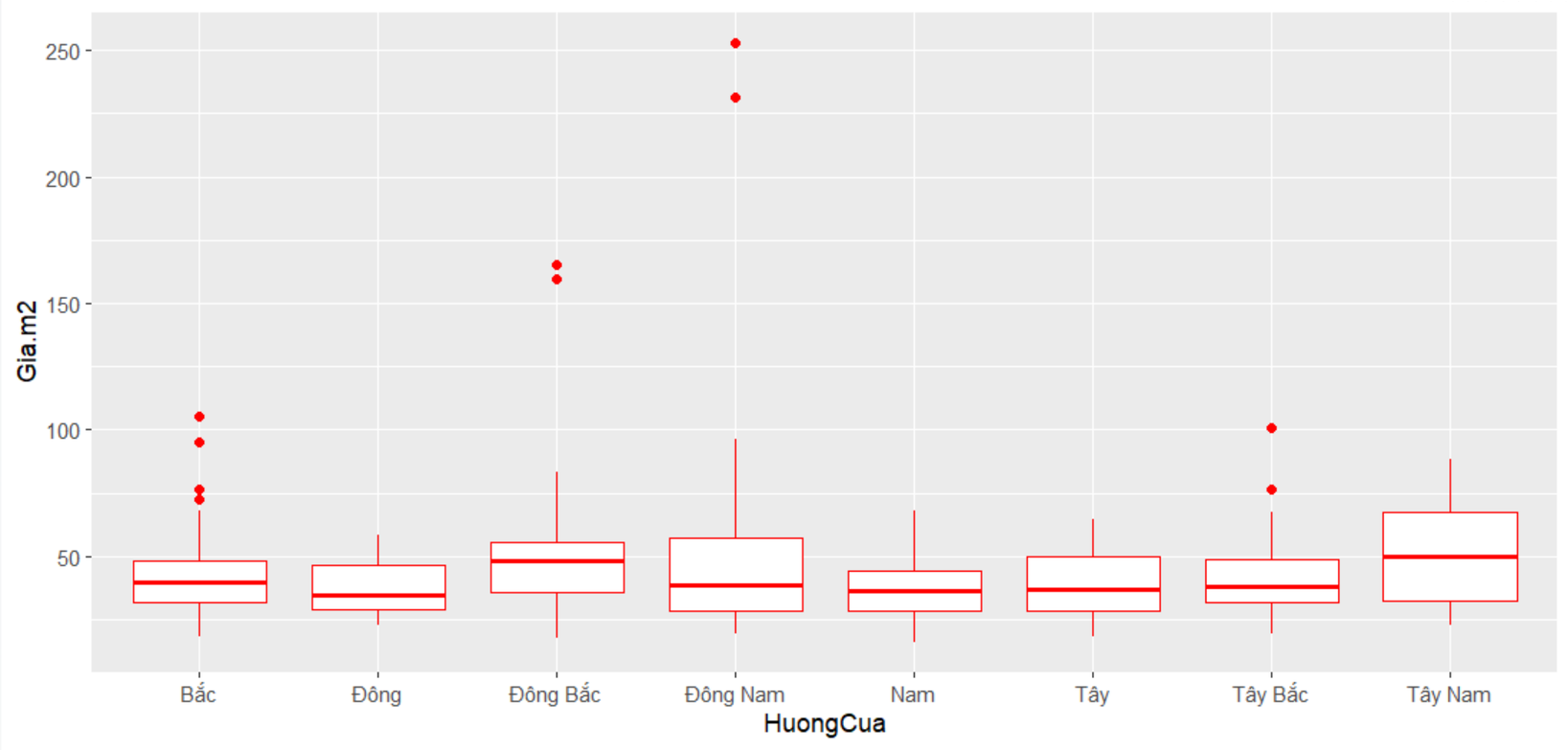
- The chart looks boring and simple
- Need to add color to make the chart look clearer and easier to read



GRAMMAR | Adding color

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_boxplot(colour="red")
```

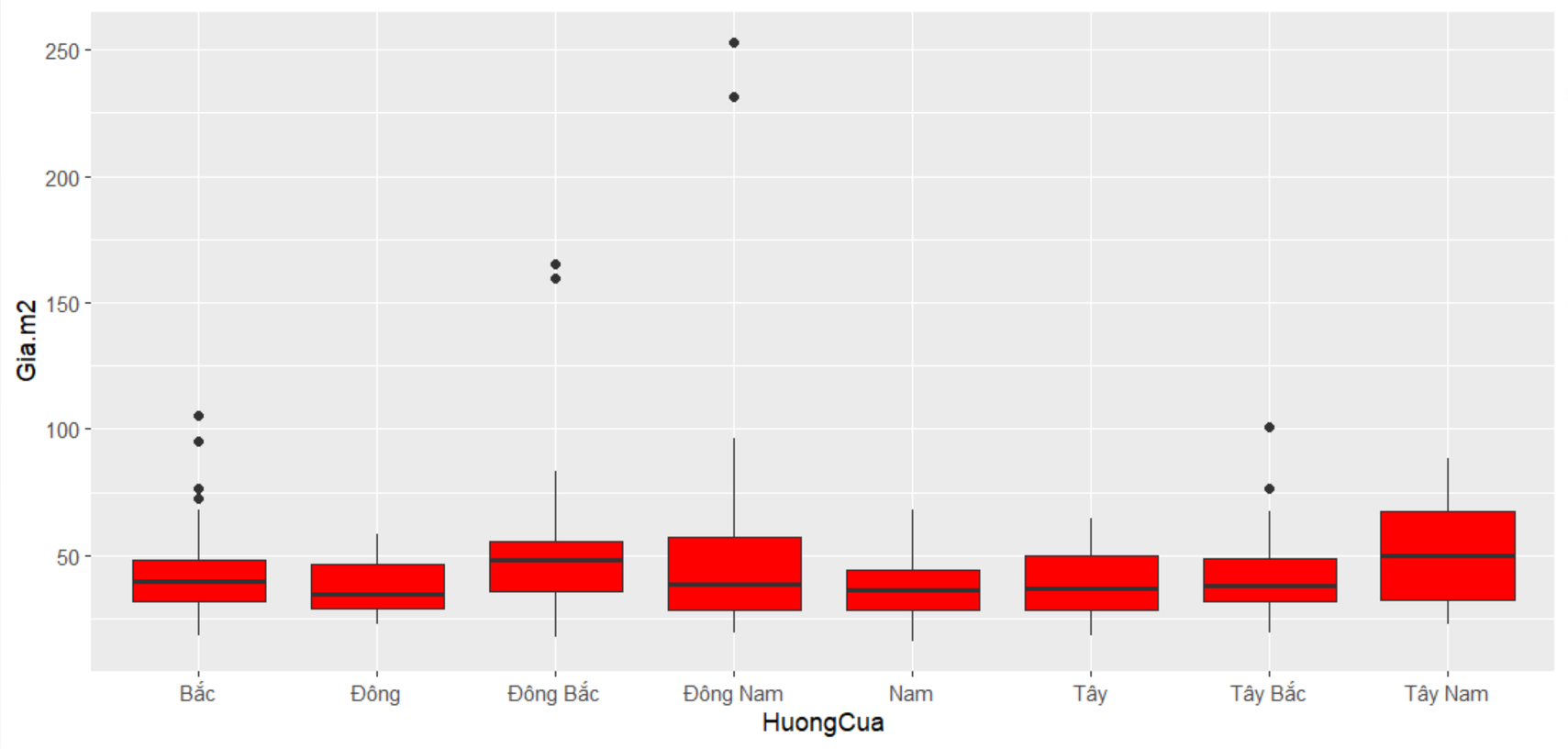
- Add color to geom function: color only the contours of the geometry



GRAMMAR | Adding color

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_boxplot(fill="red")
```

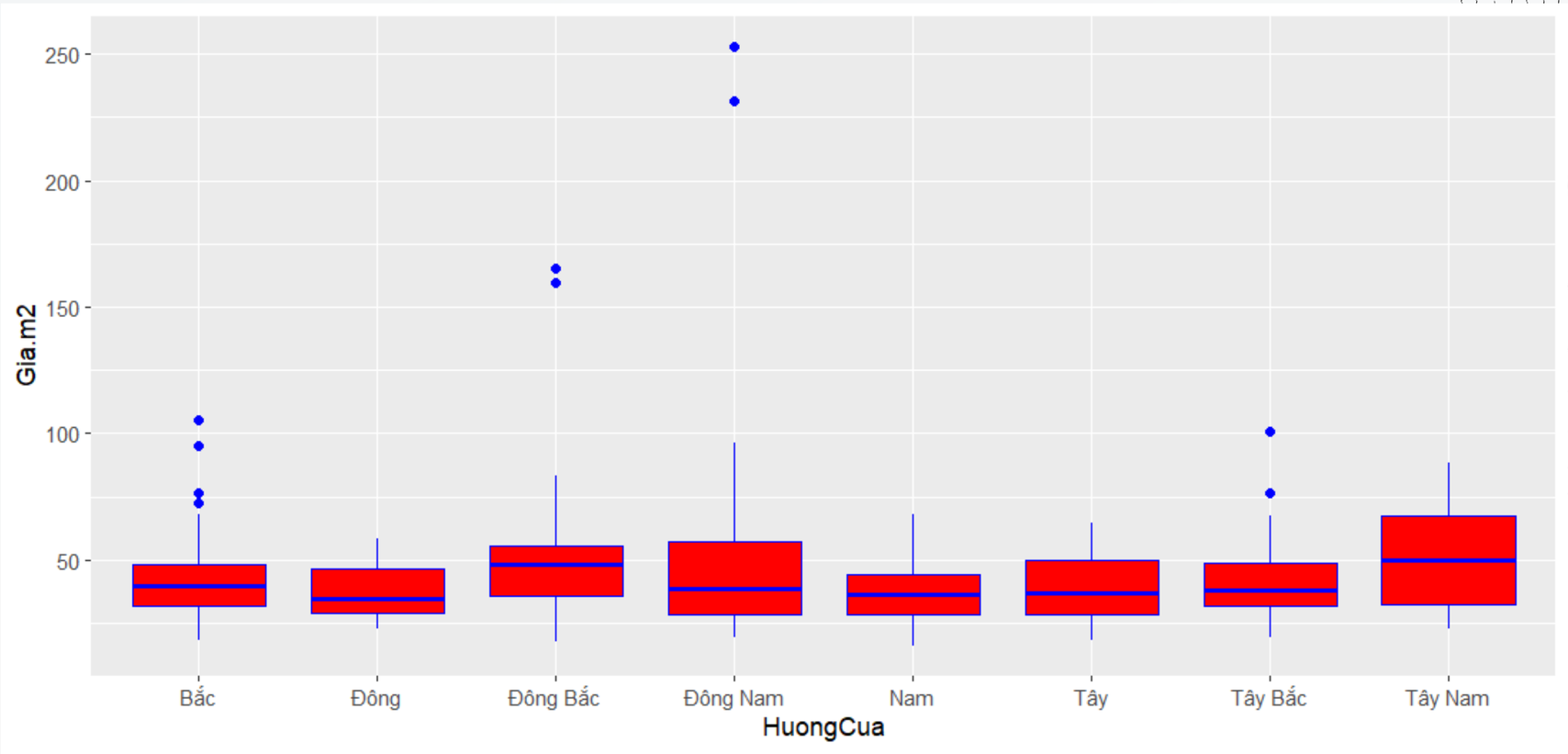
- Add color to geom function: color is filled inside the shape



GRAMMAR | Adding color

```
ggplot(data = house_df, aes(x = HuongCua, y = Gia.m2)) +
        geom_boxplot(colour="blue", fill="red")
```

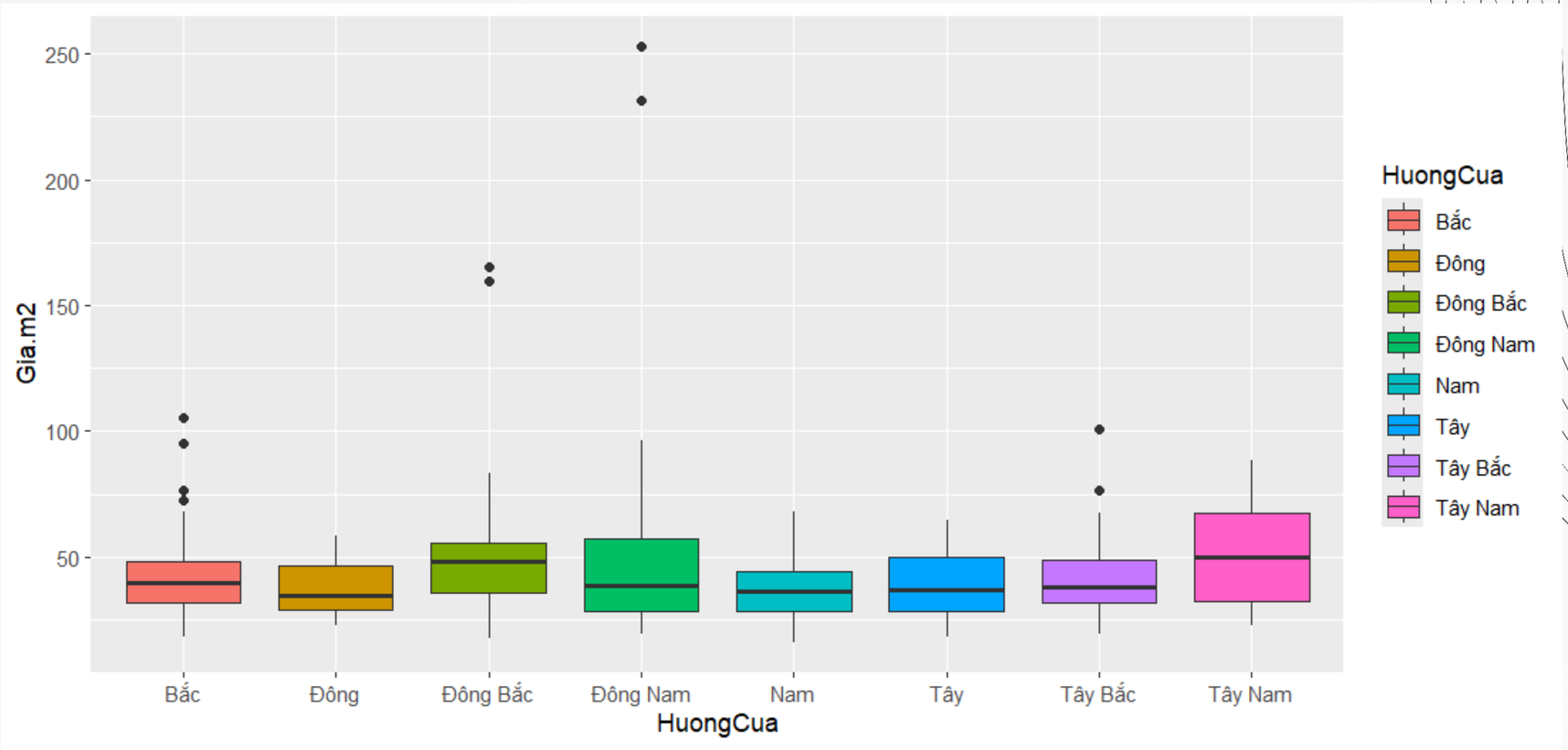
- Border color and fill color in shape



GRAMMAR | Adding color

```
ggplot(data = house_df, aes(x = HuongCua, y = GiaBan,
        fill = HuongCua))  +
        geom_boxplot()
```

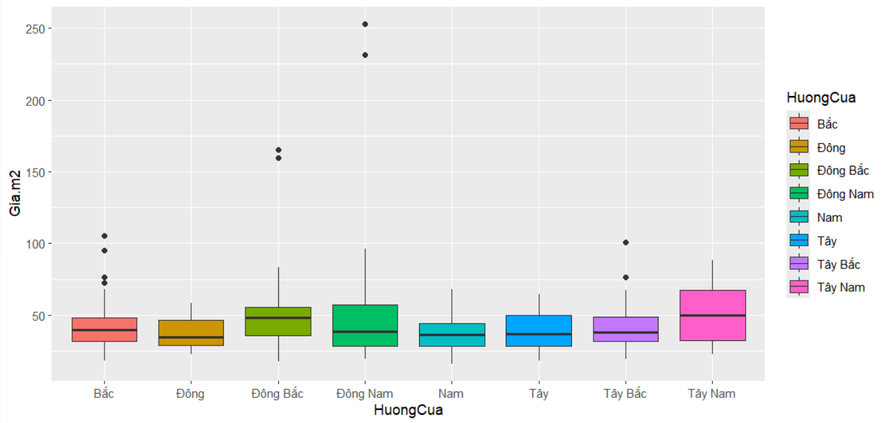
- Coloring shapes by mapping them to another column in the dataset



GRAMMAR | Scale

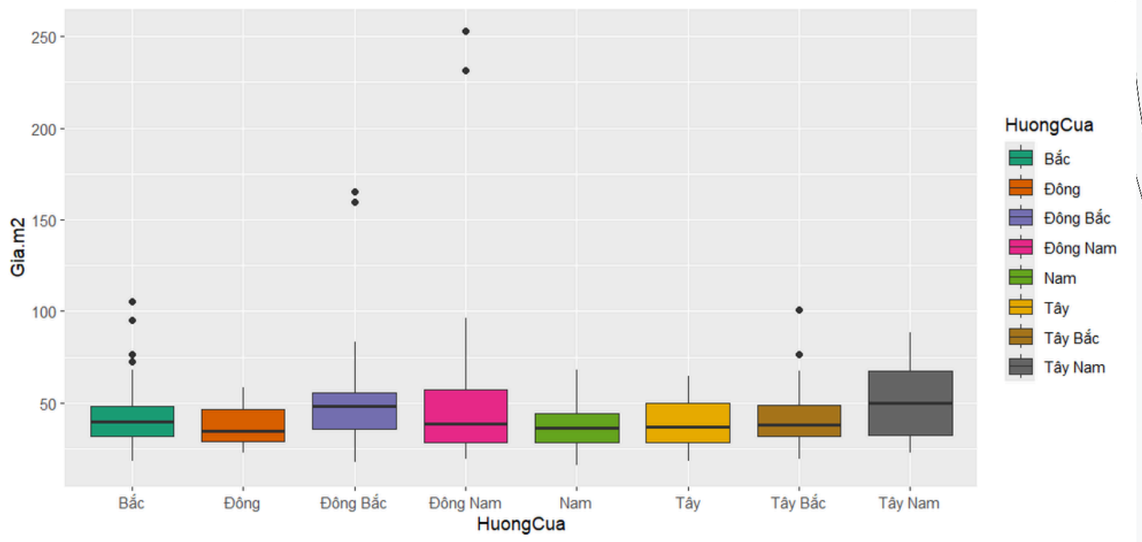
- Scale **modify an aesthetic**

base_plot

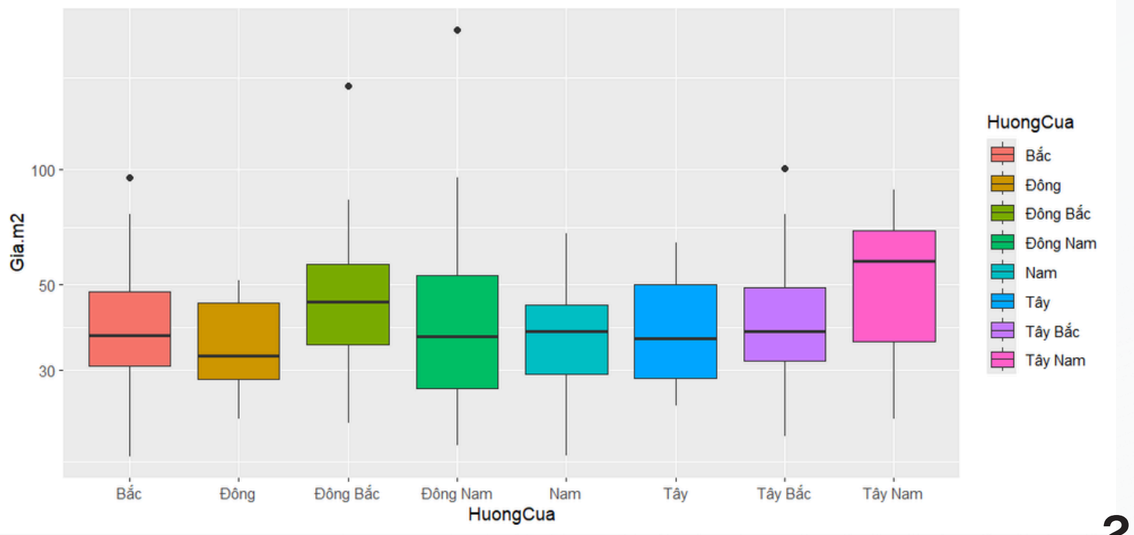


+

scale_fill_brewer(palette="Dark2")



scale_y_log10()



GRAMMAR | Facet

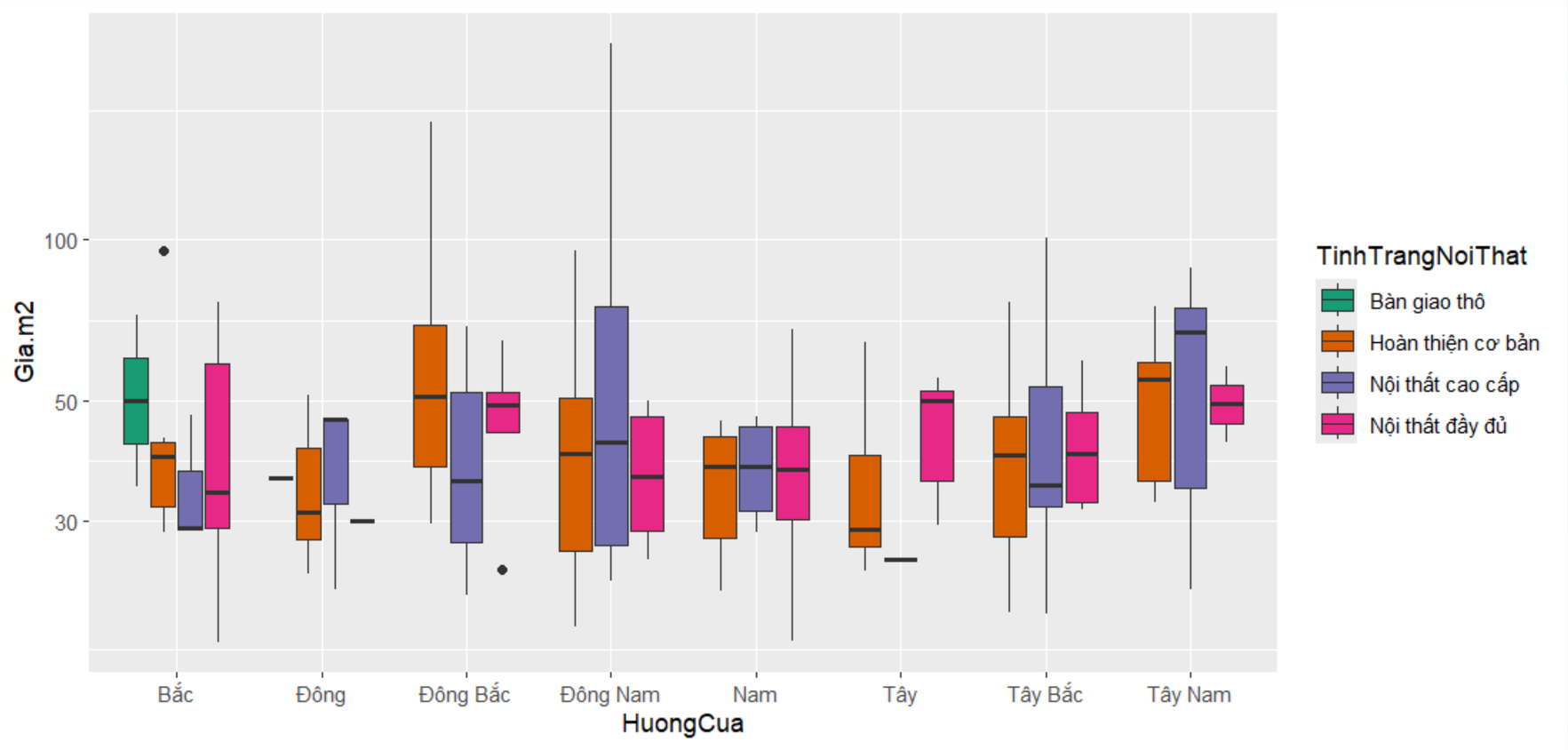
- **Facet:** creates multiple panels, each showing a specific subset of your data.
- This allows you to quickly compare patterns across different groups in your data.

Three types of faceting:

- **facet_null() (default):** Creates a single plot.
- **facet_wrap():** Arranges panels horizontally or vertically based on a variable.
- **facet_grid()**: Creates a grid of panels defined by row and column variables.

GRAMMAR | Facet

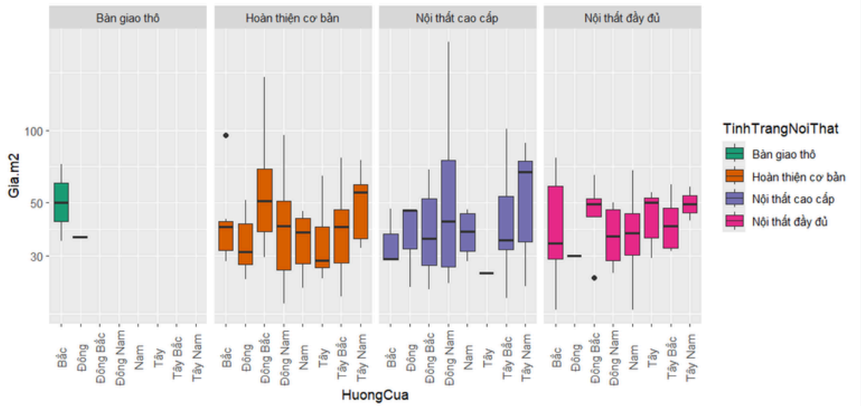
```
base_plot ← ggplot(data=house_df, aes(x=HuongCua, y=Gia.m2, fill=TinhTrangNoiThat))
+ geom_boxplot() + scale_fill_brewer(palette="Dark2") + scale_y_log10()
```



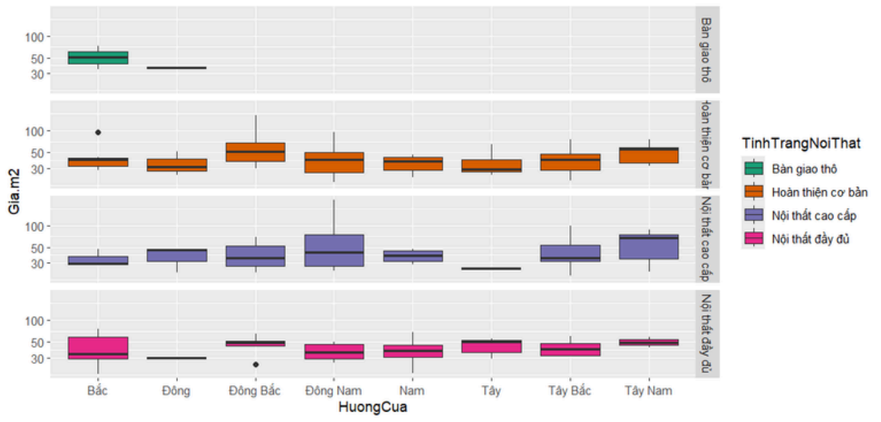
GRAMMAR | Facet

- facet_grid(~ var): Split chart vertically
- facet_grid(var ~ .): Split chart horizontally
- facet_wrap(~ var): Split and auto-edit charts

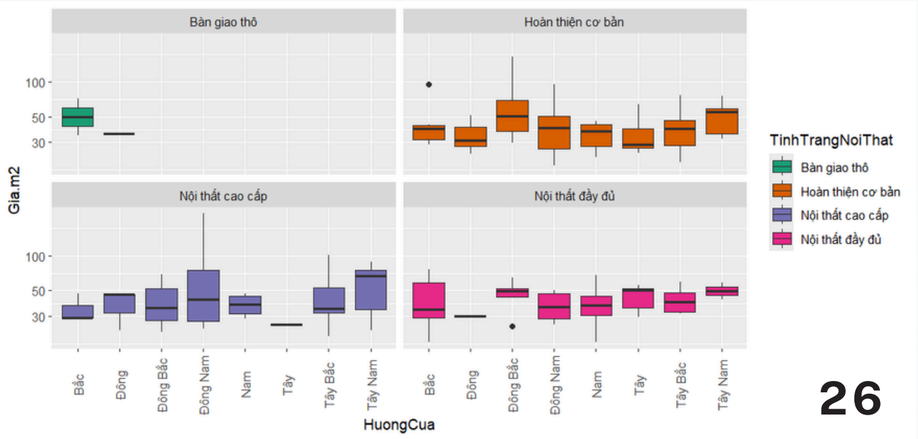
base_plot + facet_grid(~TinhTrangNoiThat)



base_plot + facet_grid(TinhTrangNoiThat~.)



base_plot + facet_wrap(~TinhTrangNoiThat)

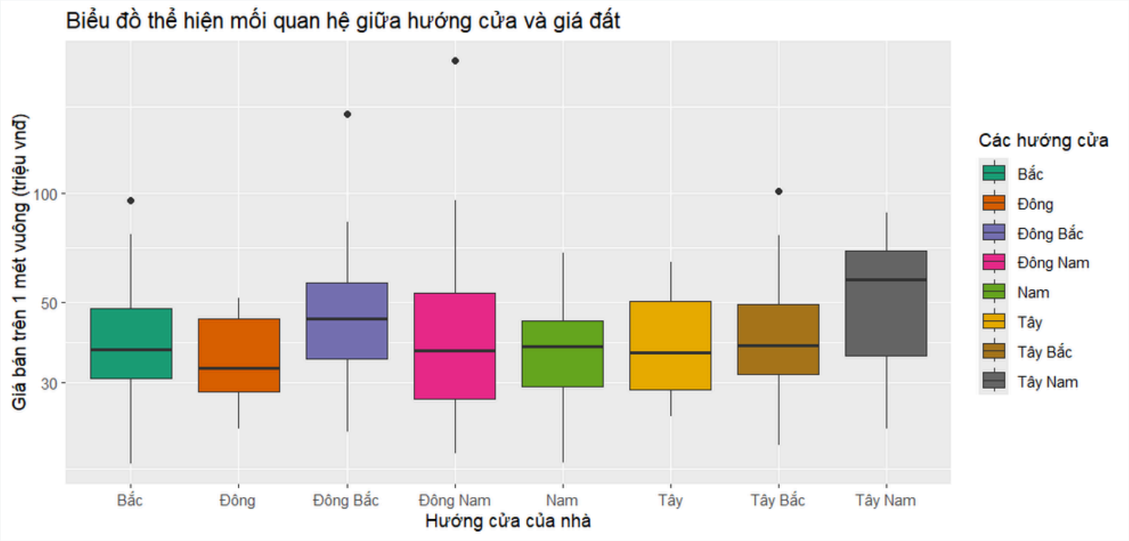


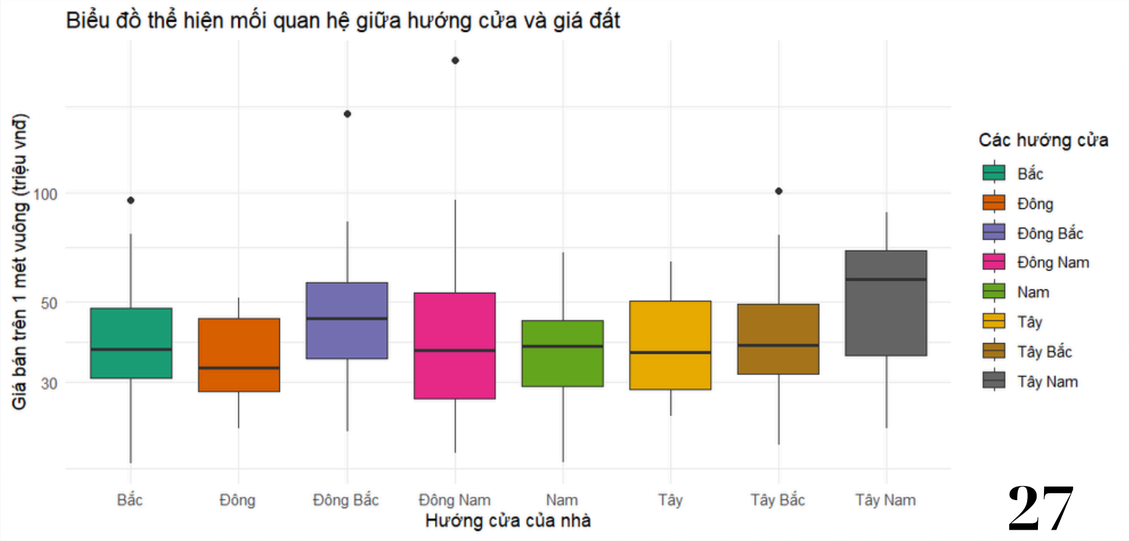
GRAMMAR | Theme

In ggplot2, **labs()** and **theme()** are two important functions that help you **customize** your chart:

- **labs():** used to set or edit chart labels, including **title, x-axis labels, y-axis labels**, and **legends**.

- **theme():** used to customize the style and appearance of a chart. It includes changing the **text style, color, size**, and layout of various chart elements, such as the **background, grid, and axes**.





LAYERS

01 INDIVIDUAL GEOMS

02 COLLECTIVE GEOMS

03 STATISTICAL SUMMARIES

04 MAPS

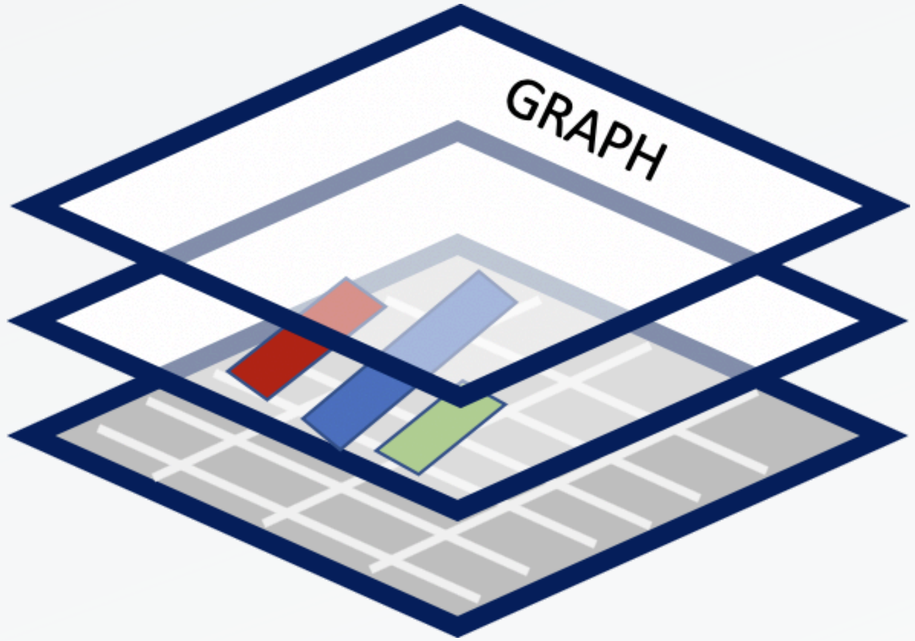
05 NETWORKS

06 ANNOTATIONS

07 ARRANGING PLOTS

PURPOSES OF LAYERS

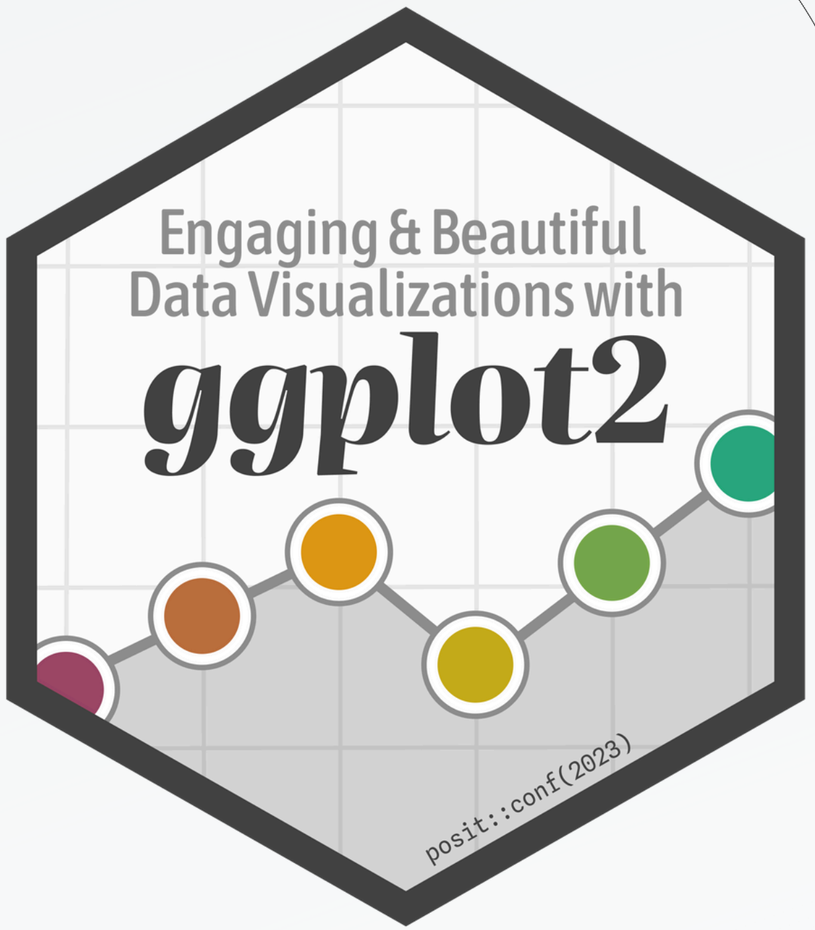
- There are 3 purposes of a layer:
  - To display the data.
  - To display a statistical summary of the data.
  - To add additional metadata: **context**, **annotations**, and **references**.



INDIVIDUAL AND COLLECTIVE GEOMS:

- An individual geom draws a distinct graphical object for each observation

- A collective geom displays multiple observations with one geometric object.

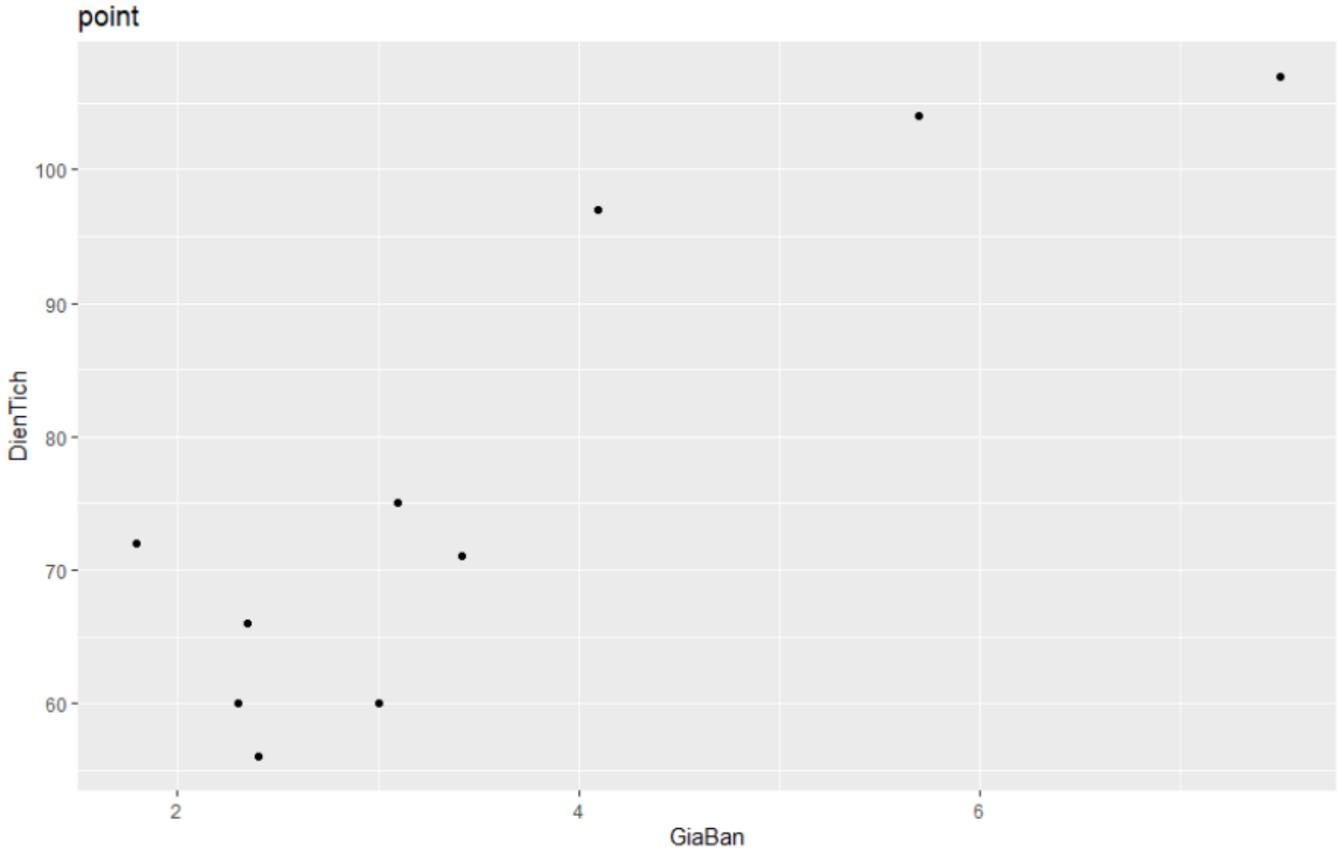


BASIC GEOMS AND MEANING

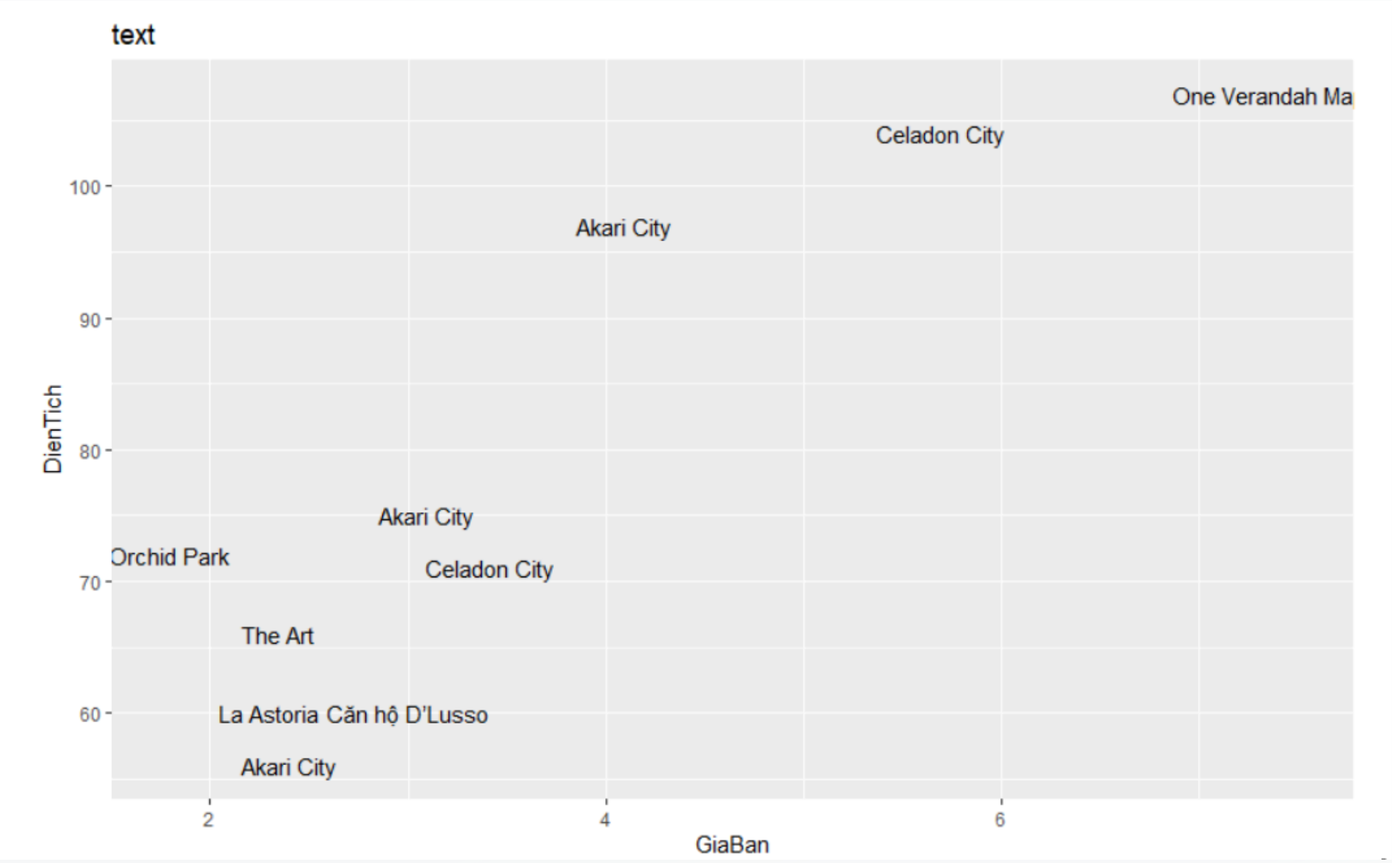
Geom	Meaning
geom_point	Plots points (used for scatter plots).
geom_line	Makes a **line plot**.
geom_bar	Makes a **bar chart**.
geom_histogram	Creates a histogram (used to display the distribution of data).
geom_text	Adds text annotations to the plot.
geom_area	Creates an area plot.
geom_label	Adds text labels with a background box.
geom_tile	Creates a heatmap (plots colored squares).
geom_polygon	Draws polygons, which are filled paths.
geom_boxplot	Creates a box plot (used to display the distribution of data).

LAYERS | Individual Geoms

```
data_subset <- head(data, 10)
p <- ggplot(data_subset, aes(GiaBan, DienTich, label = DuAn))
p + geom_point() + ggtitle("point")
p + geom_text() + ggtitle("text")
```

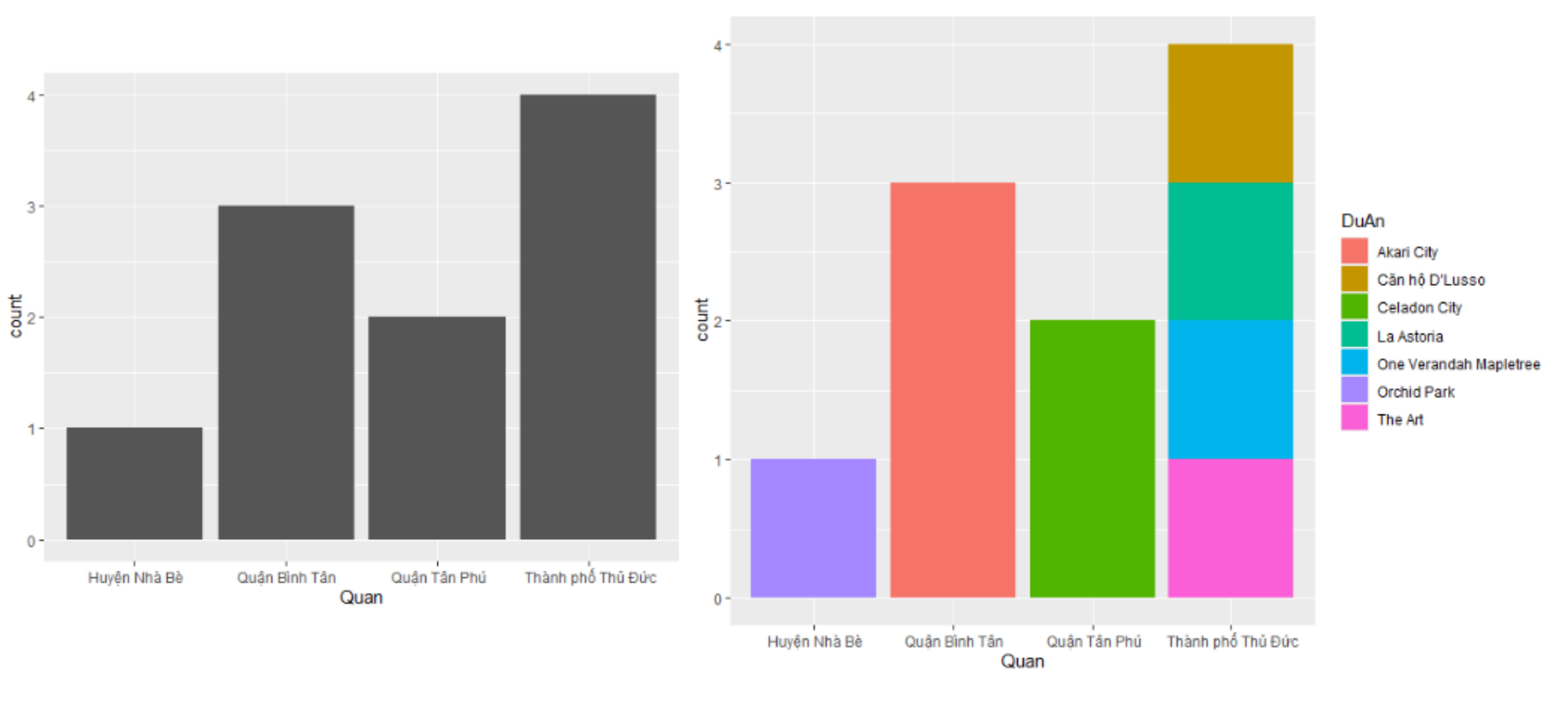


LAYERS | Individual Geoms



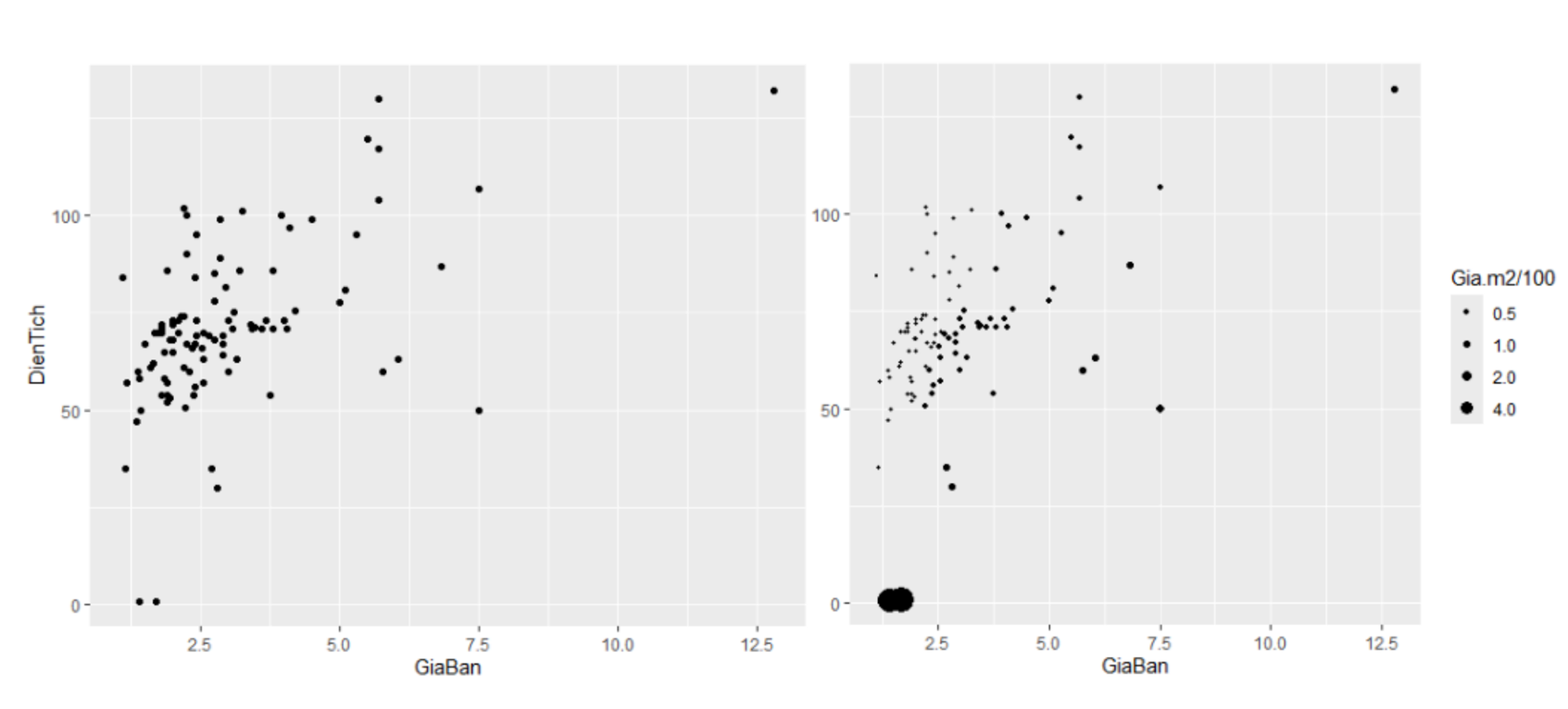
LAYERS | Collective Geoms

```
data_subset <- head(data, 10)
ggplot(data_subset, aes(Quan)) +
  geom_bar()
ggplot(data_subset, aes(Quan, fill = DuAn)) +
  geom_bar()
```



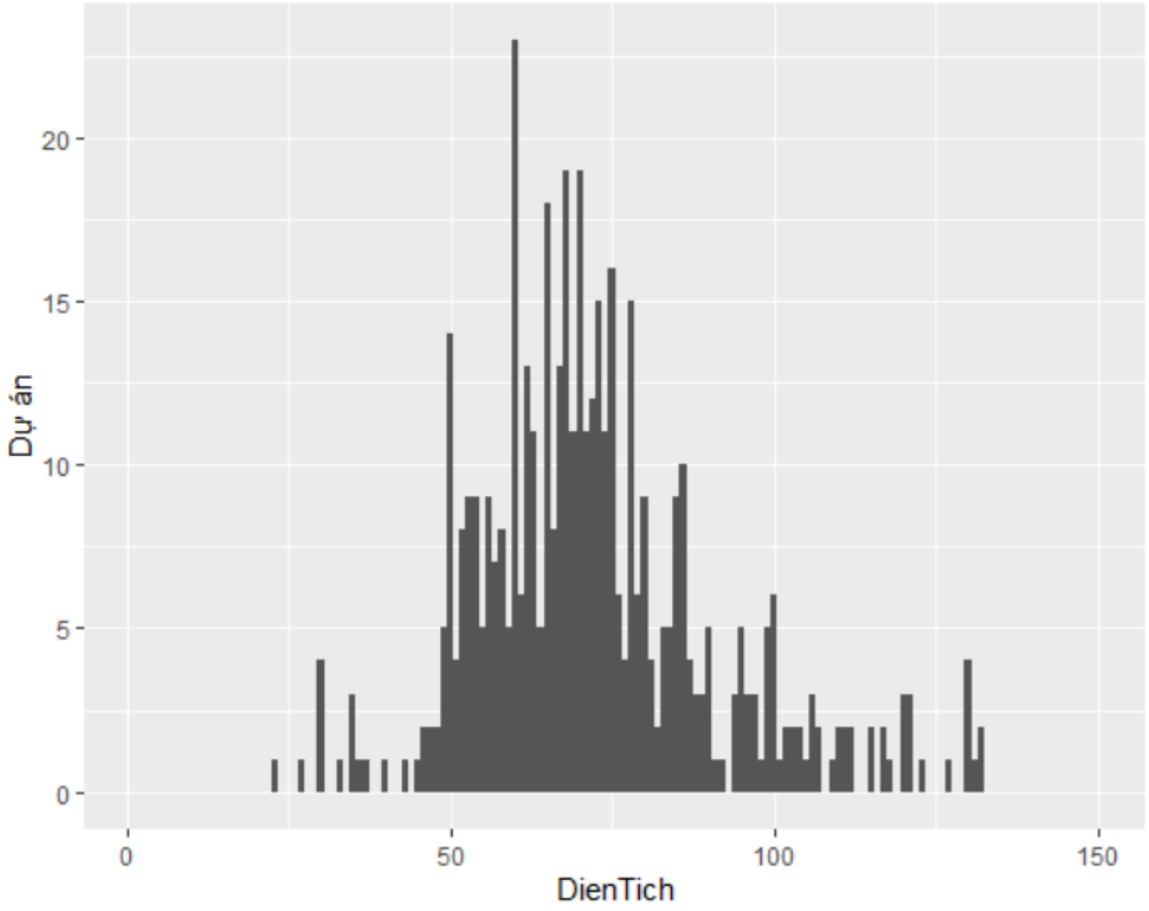
LAYERS | Statistical Summaries

```
data_subset <- head(data, 100)
ggplot(data_subset, aes(GiaBan, DienTich)) +
  geom_point()

ggplot(data_subset, aes(GiaBan, DienTich)) +
  geom_point(aes(size = Gia.m2 / 100)) +
  scale_size_area(breaks = c(0.5, 1, 2, 4))
```



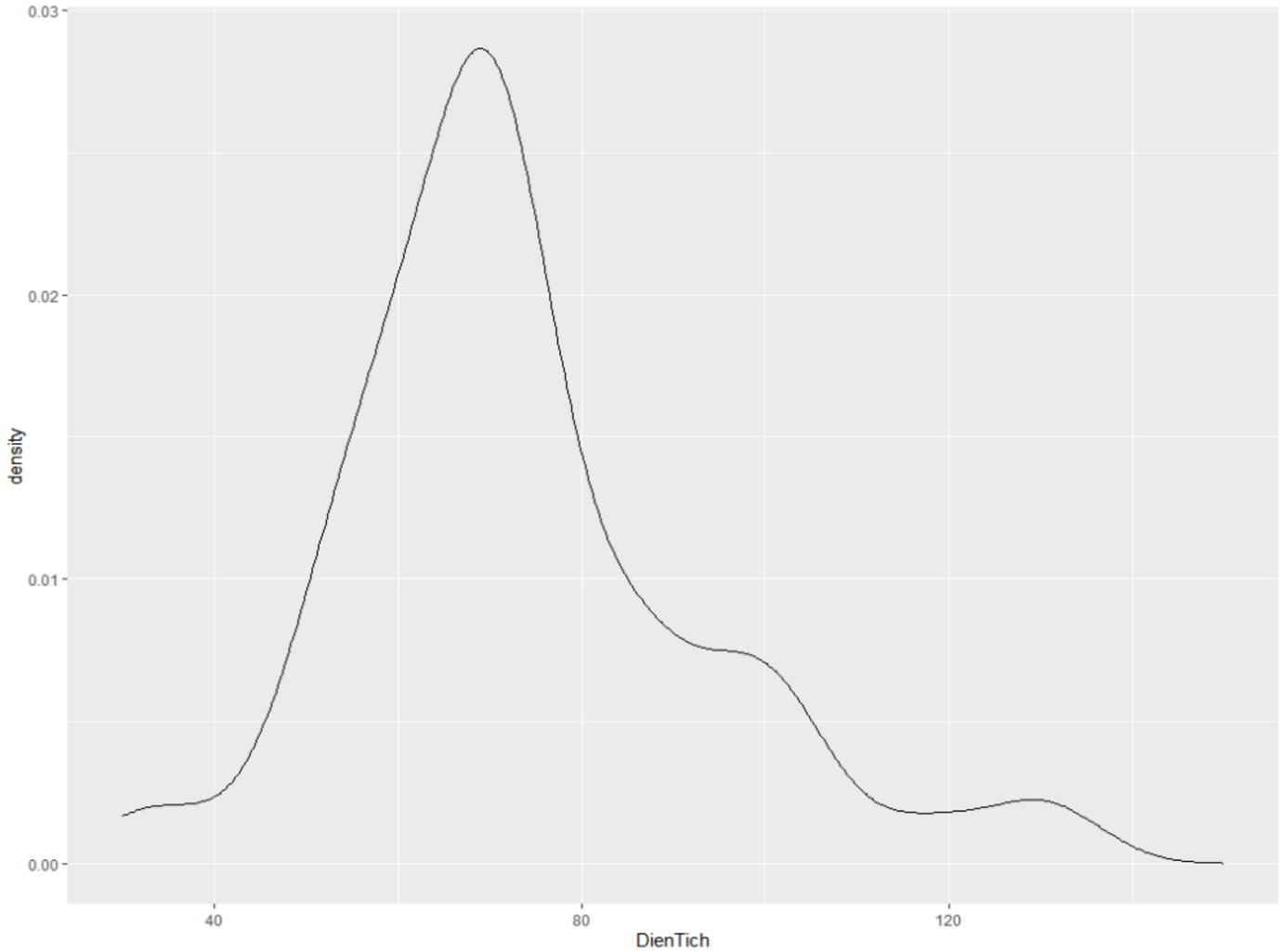
LAYERS | Statistical Summaries

```
ggplot(data_subset, aes(DienTich)) +
  geom_histogram(binwidth = 1) +
  xlim(1, 150) +
  ylab("Dự án")
```



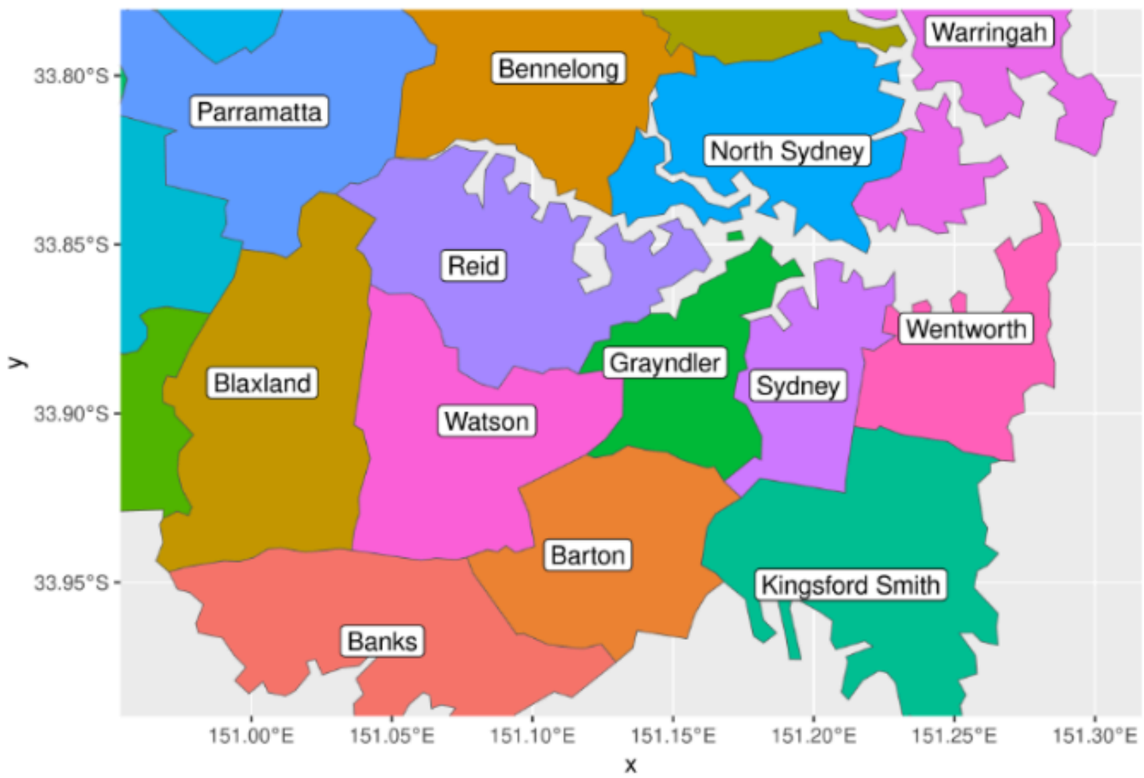
LAYERS | Statistical Summaries

```
ggplot(data_subset, aes(DienTich)) +
  geom_density(na.rm = TRUE) +
  xlim(30, 150) +
  theme(legend.position = "none")
```



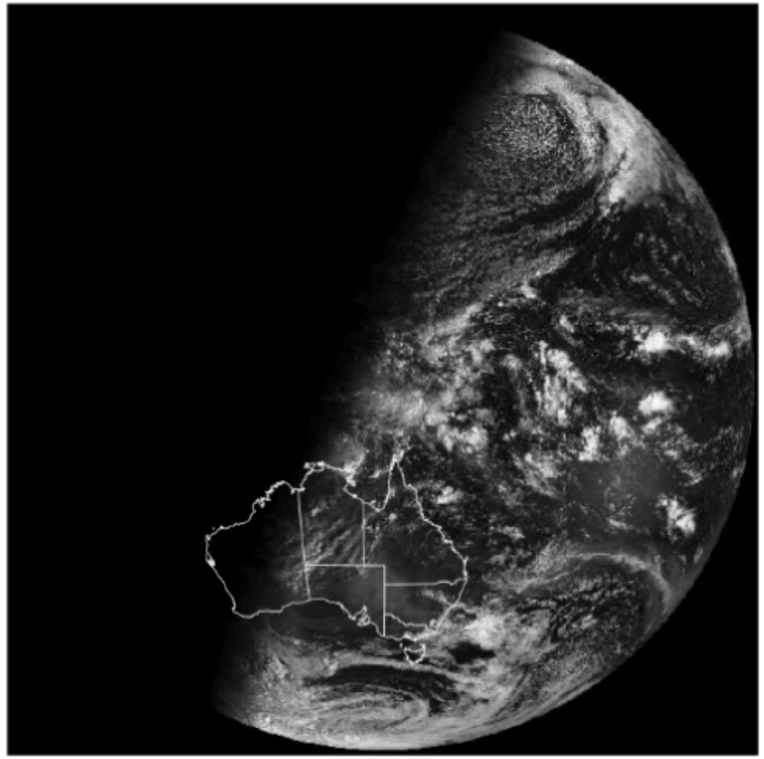
LAYERS | Maps

```r
# Draw the electoral map of Sydney
ggplot(sydney_map) +
  geom_sf(aes(fill = NAME), show.legend = FALSE) +
  coord_sf(xlim = c(150.97, 151.3), ylim = c(-33.98, -33.79)) +
  geom_sf_label(aes(label = NAME), label.padding = unit(1, "mm"))
#> Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
#> give correct results for longitude/latitude data
```



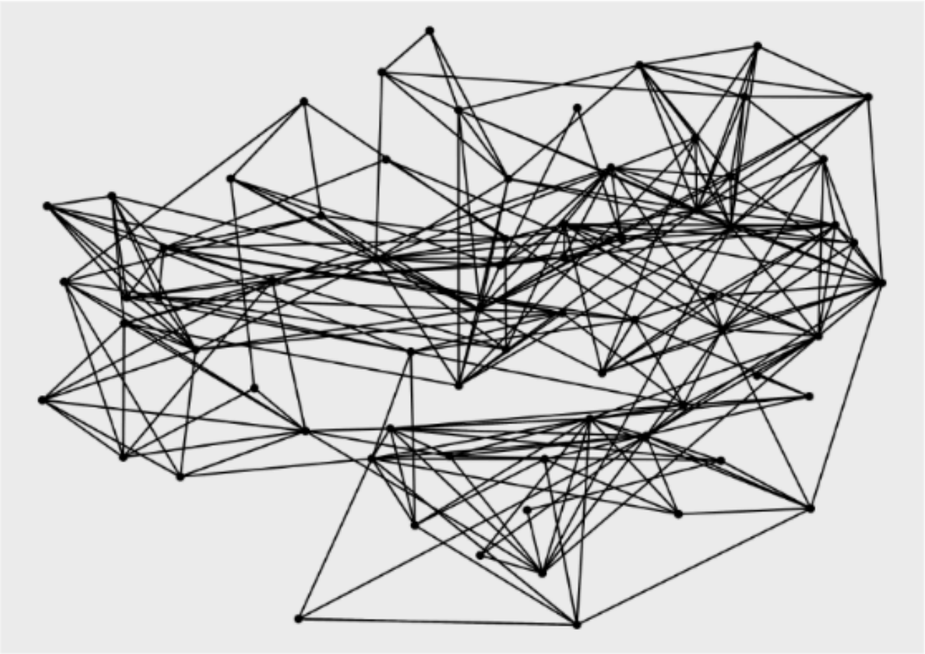
LAYERS | Maps

```
ggplot() +
  geom_stars(data = sat_vis, show.legend = FALSE) +
  geom_sf(data = oz_states, fill = NA, color = "white") +
  coord_sf() +
  theme_void() +
  scale_fill_gradient(low = "black", high = "white")
```

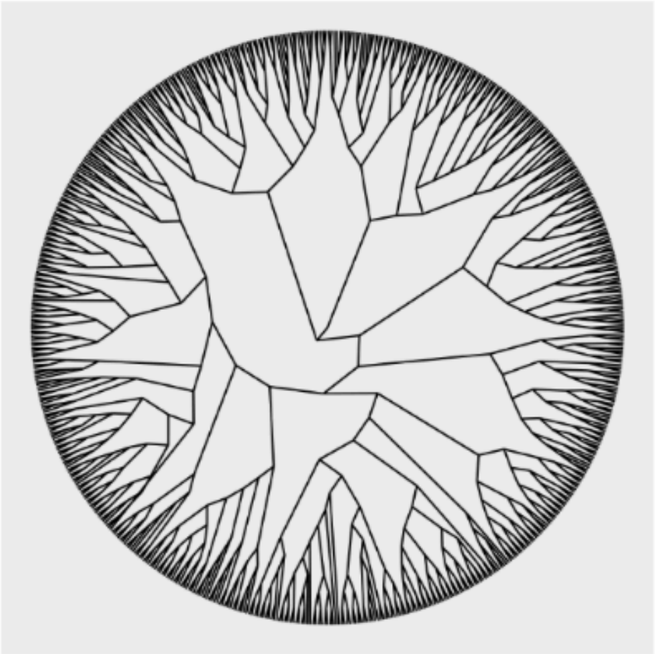


LAYERS | Networks

```
ggraph(hs_graph, layout = "drl") +
  geom_edge_link() +
  geom_node_point()
```

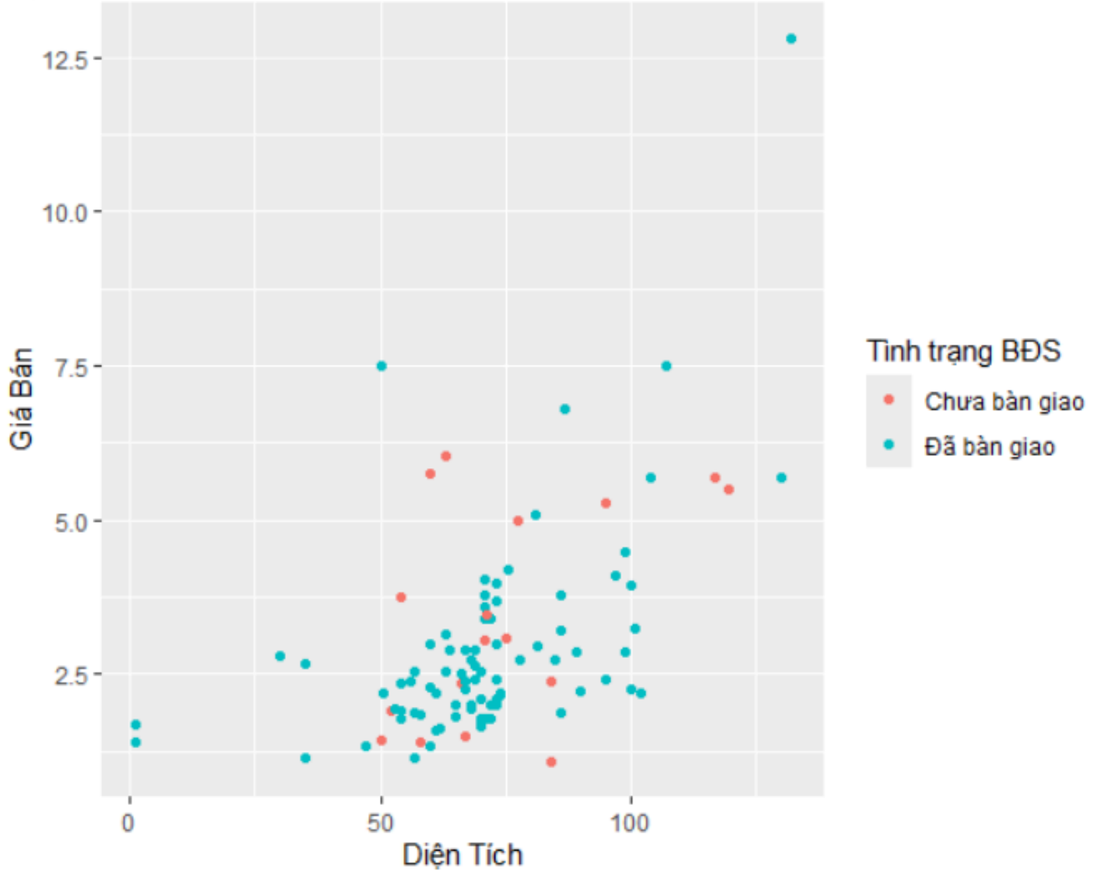


```
ggraph(luv_graph, layout = "dendrogram", circular = TRUE) +
  geom_edge_link() +
  coord_fixed()
```



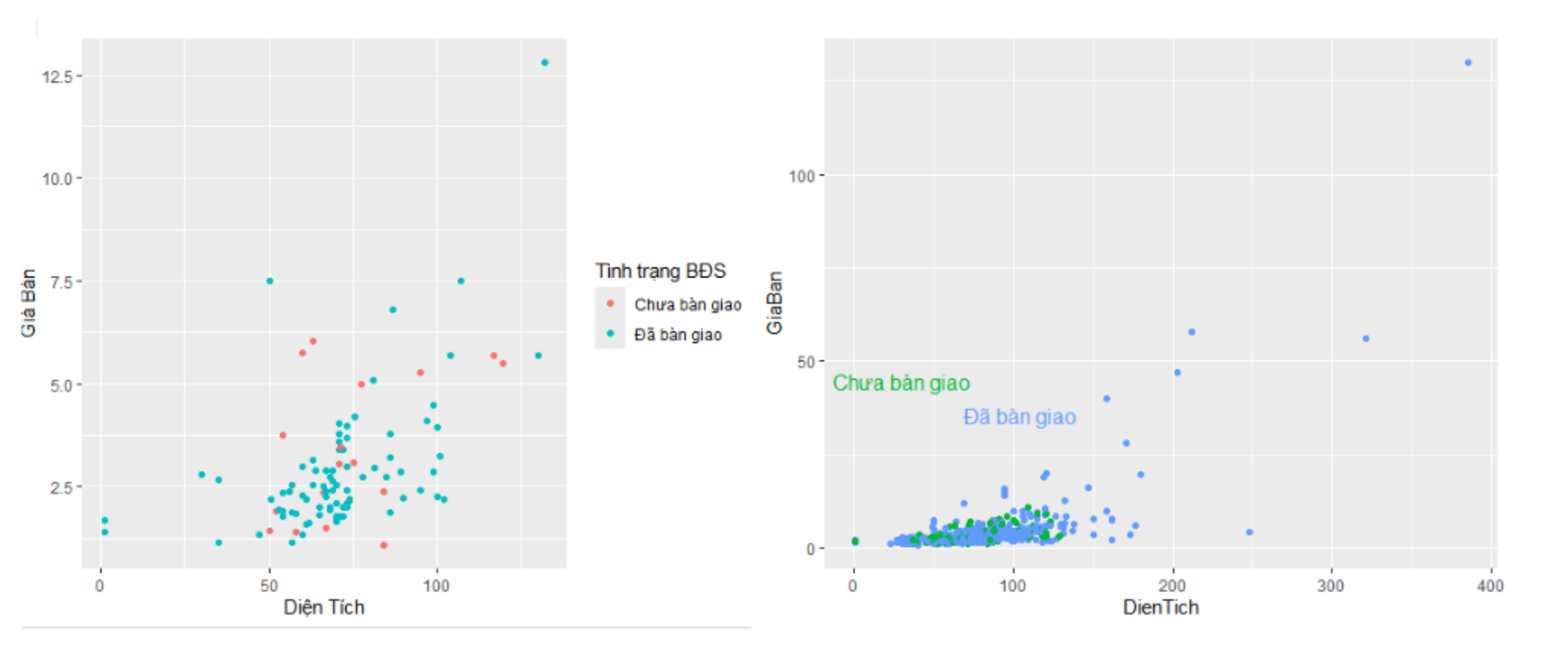
LAYERS | Annotations

```
data_subset <- head(data,100)
ggplot(data_subset, aes(DienTich, GiaBan)) +
  geom_point(aes(colour = TinhTrangBDS)) +
  labs (
    x = "Diện Tích",
    y = "Giá Bán",
    colour = "Tình trạng BĐS"
  )
```



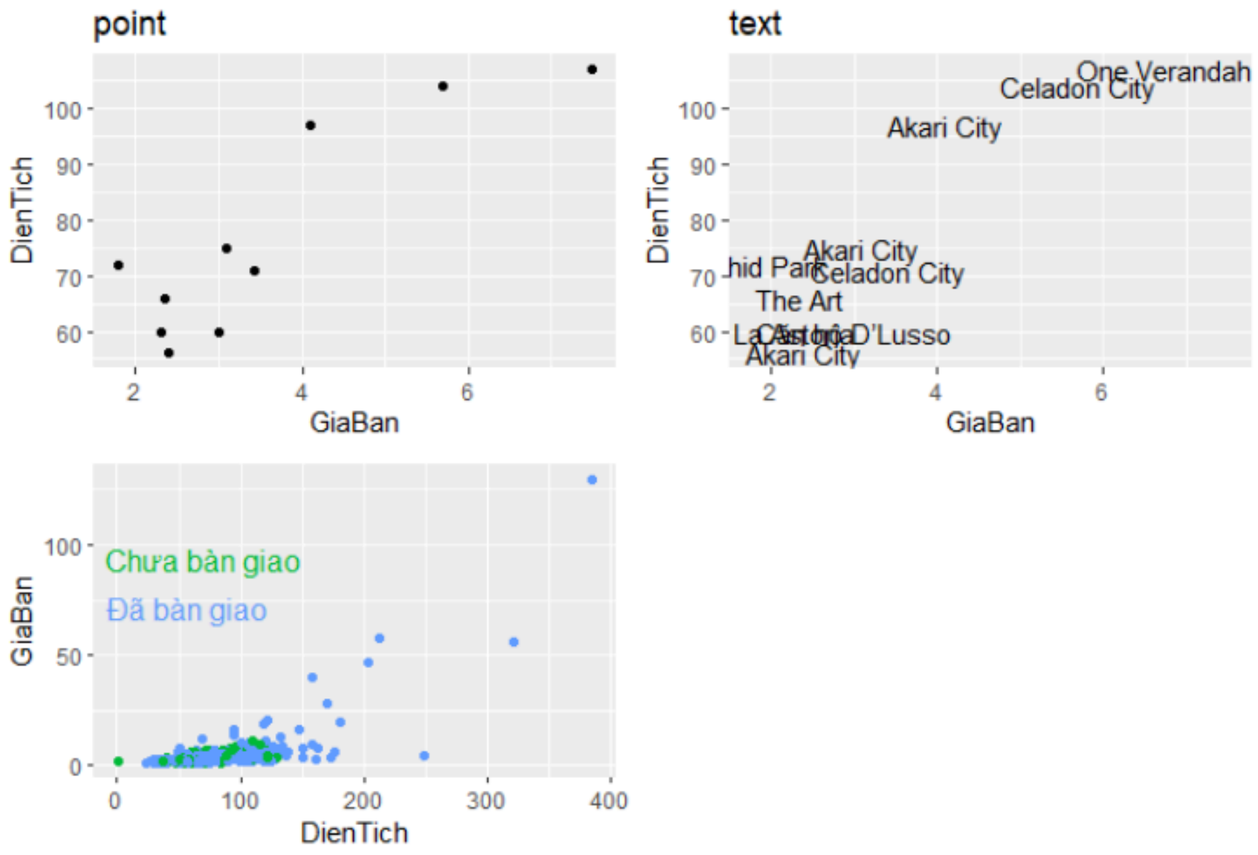
LAYERS | Annotations

```
p3 <- ggplot(data_subset, aes(DienTich, GiaBan, colour = TinhTrangBDS)) +
  geom_point(show.legend = FALSE) +
  directlabels::geom_dl(aes(label = TinhTrangBDS), method = "smart.grid")
```

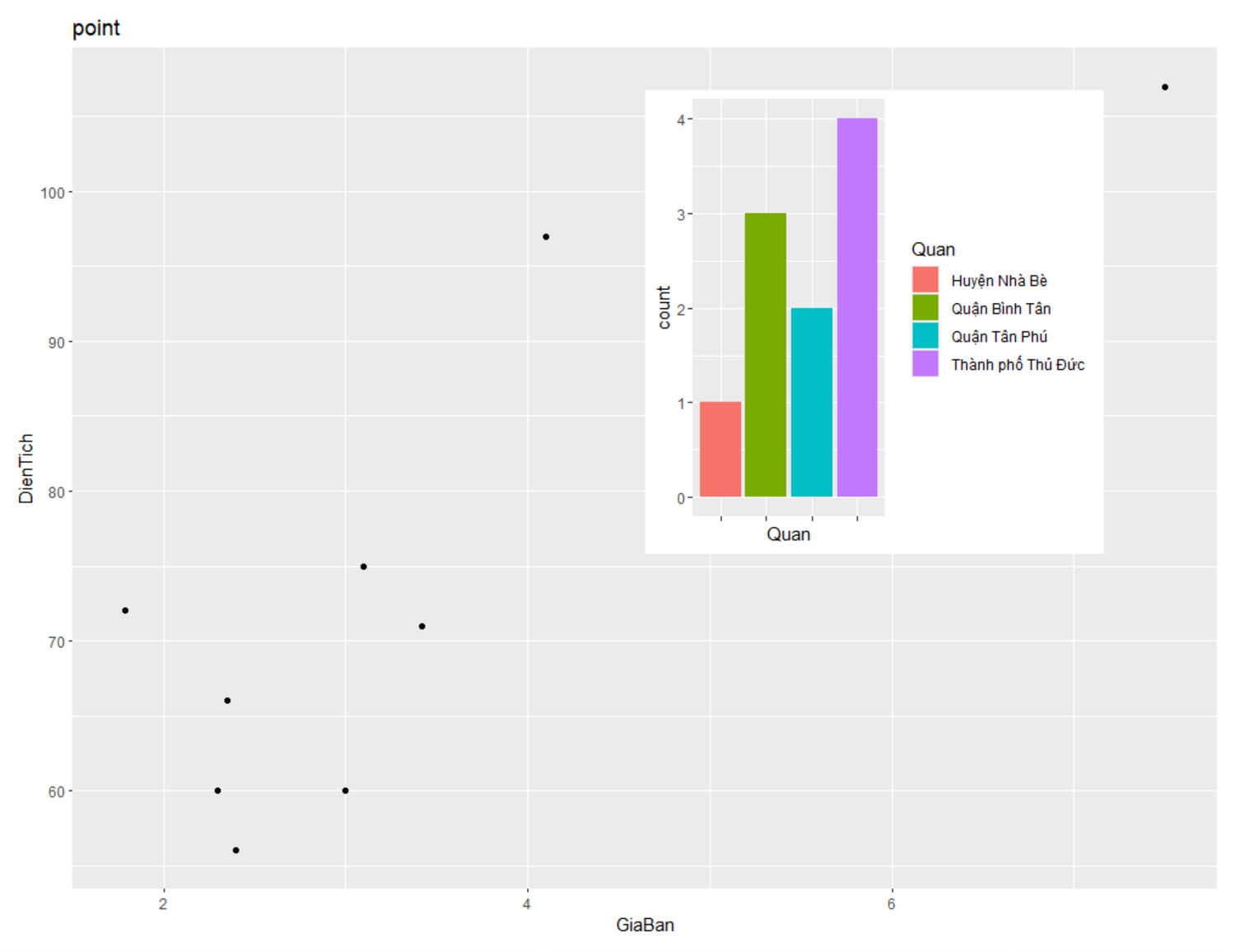


LAYERS | Arranging plots

```
p1 + p2 + p3 + plot_layout(ncol = 2)
```



LAYERS | Arranging plots



SCALE

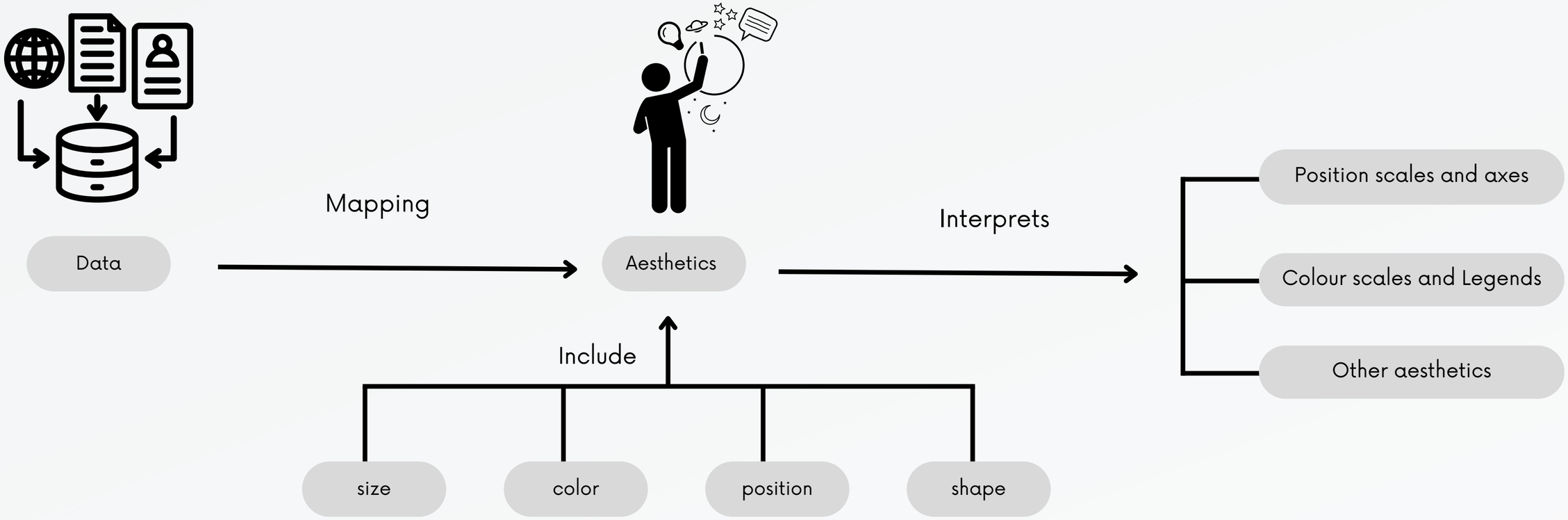
01 SOME INFORMATION

02 POSITIONS SCALE AND AXES

03 COLOR SCALES AND LEGENDS

04 OTHER AESTHETICS

SOME INFORMATION



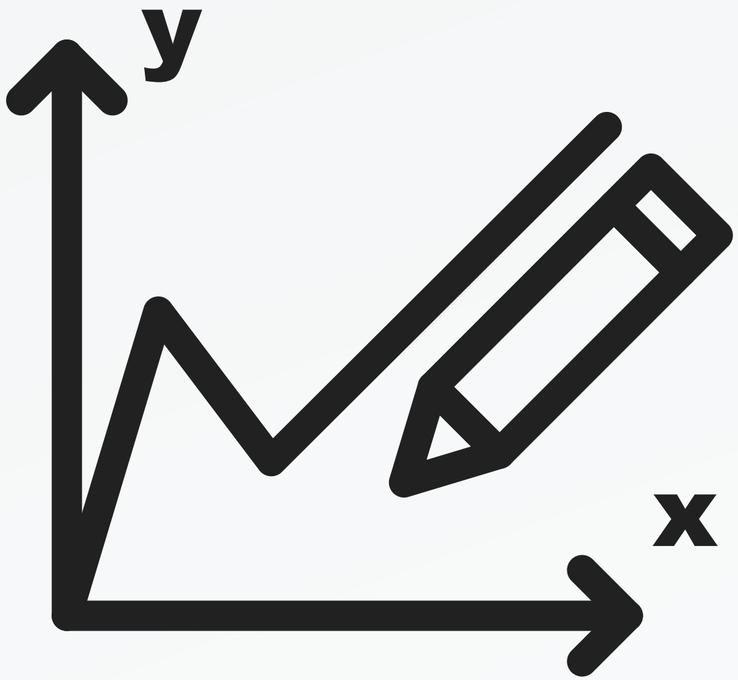
POSITION SCALES AND AXES

Used to control the locations of visual entities in a plot.

Every plot has two position scales, corresponding to the x and y aesthetics.

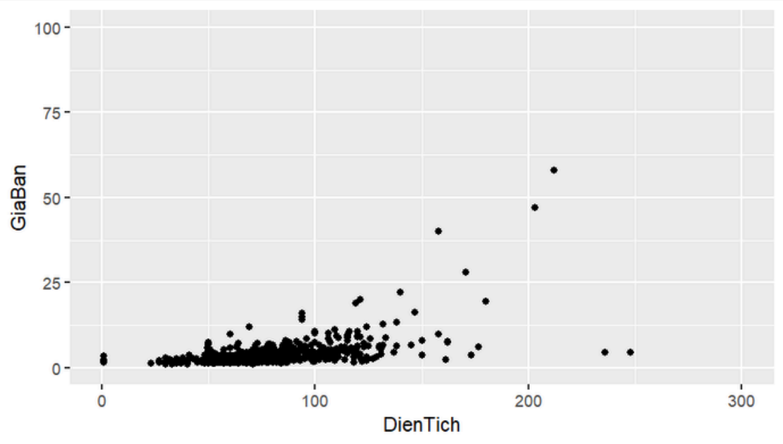
Let's delve into the four primary types of position scales to understand their application with various data types:

1. **Numeric (Continuous) Position Scales**
2. **Date/Time Scales**
3. **Discrete Position Scales**
4. **Binned Position Scales**



POSITION SCALES AND AXES | LIMITS + ZOOMING IN

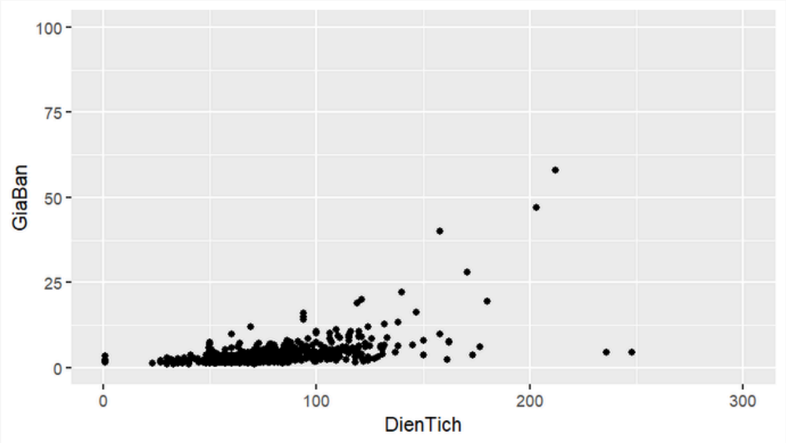
- **Scales Have Limits:** Every chart has a range for the data it displays.
- **Consistency Across Plots:** When comparing multiple plots, it's crucial to manually set consistent scale limits to facilitate accurate visual comparisons.
- **Data Truncation Warning:** Adjusting limits can exclude data, turning it into "NA" and possibly distorting the plot.




```
ggplot(house, aes(DienTich, GiaBan)) + geom_point() + xlim(0, 300) + ylim(0, 100)
```

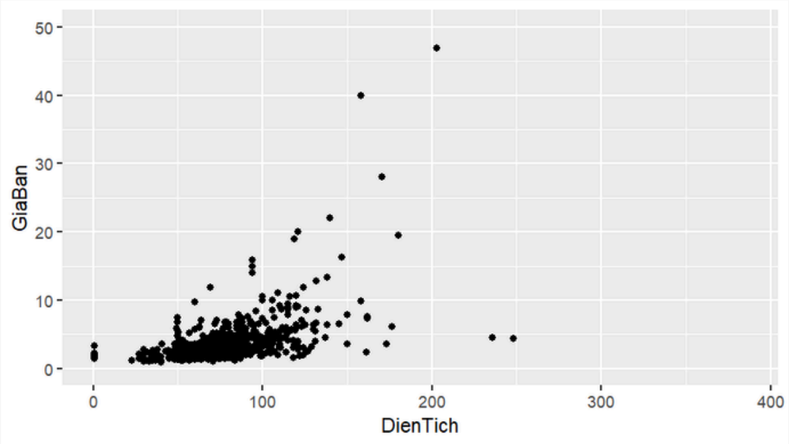
POSITION SCALES AND AXES | LIMITS + ZOOMING IN

- **Scales Have Limits:** Every chart has a range for the data it displays.
- **Consistency Across Plots:** When comparing multiple plots, it's crucial to manually set consistent scale limits to facilitate accurate visual comparisons.
- **Data Truncation Warning:** Adjusting limits can exclude data, turning it into "NA" and possibly distorting the plot.



ggplot(house, aes(DienTich, GiaBan)) + geom_point() + xlim(0, 300) + ylim(0, 100)

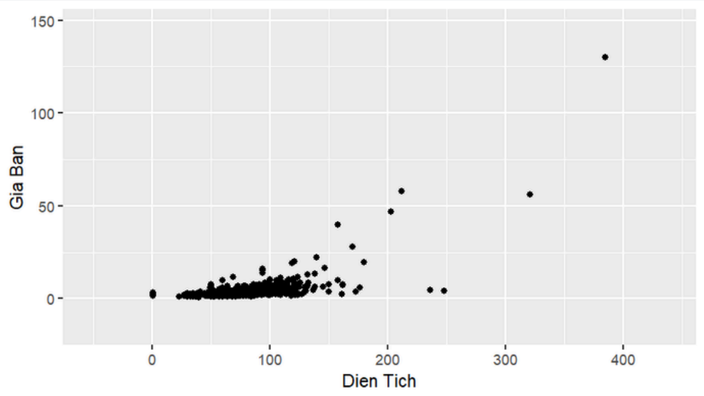
ggplot(house, aes(DienTich, GiaBan)) + geom_point() + coord_cartesian(ylim = c(0, 50))



POSITION SCALES AND AXES | VISUAL RANGE EXPANSION

- **Custom Expansion**: Override default expansion using expand = expansion() in scale_x_continuous() and scale_y_continuous().
- **Additive and Multiplicative Factors**:
  - Additive: Adds a constant space.
  - Multiplicative: Adds space proportional to axis size.

```
base <- ggplot(house, aes(DienTich, GiaBan)) +geom_point() +
theme(legend.position = "none") + labs(x = "Dien Tich", y = "Gia Ban")
```

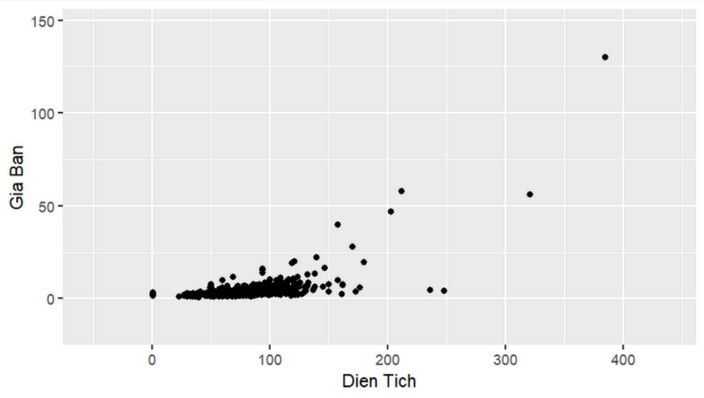



```
base +

scale_x_continuous(expand
= expansion(mult = .2)) +

scale_y_continuous(expand
= expansion(mult = .2))
```

POSITION SCALES AND AXES | VISUAL RANGE EXPANSION

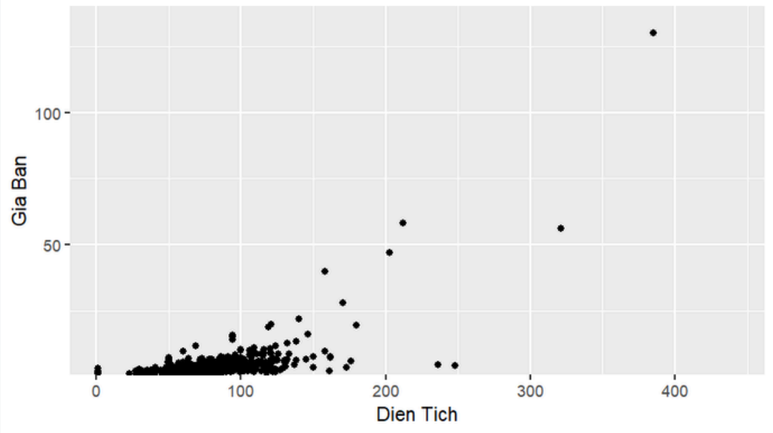
- **Custom Expansion**: Override default expansion using expand = expansion() in scale_x_continuous() and scale_y_continuous().
- **Additive and Multiplicative Factors**:
  - Additive: Adds a constant space.
  - Multiplicative: Adds space proportional to axis size.

```
base <- ggplot(house, aes(DienTich, GiaBan)) +geom_point() +
theme(legend.position = "none") + labs(x = "Dien Tich", y = "Gia Ban")
```




```
base +

scale_x_continuous(expand
= expansion(mult = .2)) +

scale_y_continuous(expand
= expansion(mult = .2))
```

```
base +
scale_x_continuous(expand
= expansion(mult = c(.05,
.2))) +
scale_y_continuous(expand
= expansion(add = c(0, 10)))
```



POSITION SCALES AND AXES | BREAKS + MINOR BREAKS + LABELS

Breaks:

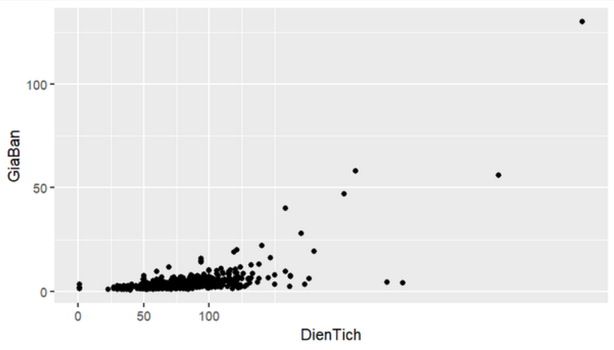
- **Axis Control:** Use the breaks argument to set precise axis tick marks.
- **Manual Settings**: Define breaks with specific values or remove with NULL.

Minor Breaks:

- **Faint Grid Lines:** Add subtle grid lines between major ticks.
- **Highlight Non-Linearity:** Useful for log scales.

Labels:

- **Setting Labels**: Assign text to breaks using the labels argument.
- **Removing Labels:** Use labels = NULL to remove axis or legend labels but keep breaks.
- **Handling Overlap:** Even with ample space, axis labels may overlap, requiring careful customization.



base + scale_x_continuous(breaks = c(0, 50, 100))

POSITION SCALES AND AXES | BREAKS + MINOR BREAKS + LABELS

Breaks:

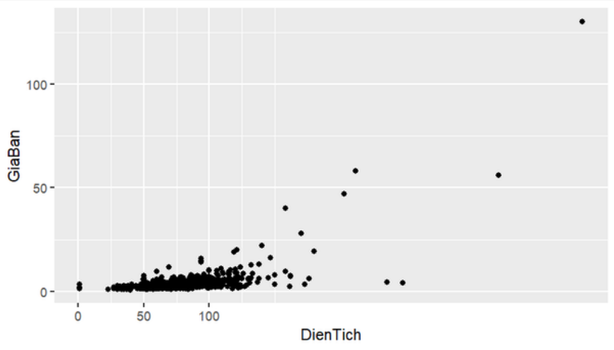
- **Axis Control:** Use the breaks argument to set precise axis tick marks.
- **Scale Breaks:** Control both axis and legend tick marks.
- **Manual Settings**: Define breaks with specific values or remove with NULL.

Minor Breaks:

- **Faint Grid Lines:** Add subtle grid lines between major ticks.
- **Highlight Non-Linearity:** Useful for log scales.

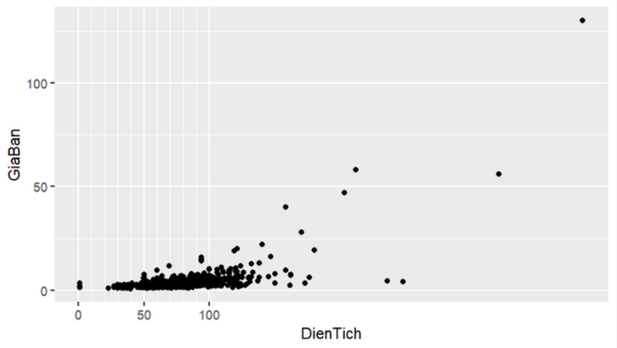
Labels:

- **Setting Labels**: Assign text to breaks using the labels argument.
- **Removing Labels:** Use labels = NULL to remove axis or legend labels but keep breaks.
- **Handling Overlap:** Even with ample space, axis labels may overlap, requiring careful customization.



base + scale_x_continuous(breaks = c(0, 50, 100))

base + scale_x_continuous(breaks = c(0, 50, 100), minor_breaks = seq(0, 100, 10))



POSITION SCALES AND AXES | BREAKS + MINOR BREAKS + LABELS

Breaks:

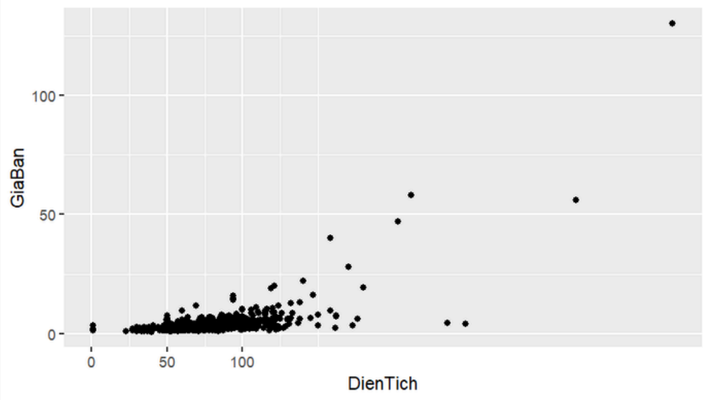
- **Axis Control:** Use the breaks argument to set precise axis tick marks.
- **Scale Breaks:** Control both axis and legend tick marks.
- **Manual Settings**: Define breaks with specific values or remove with NULL.

Minor Breaks:

- **Faint Grid Lines:** Add subtle grid lines between major ticks.
- **Highlight Non-Linearity:** Useful for log scales.

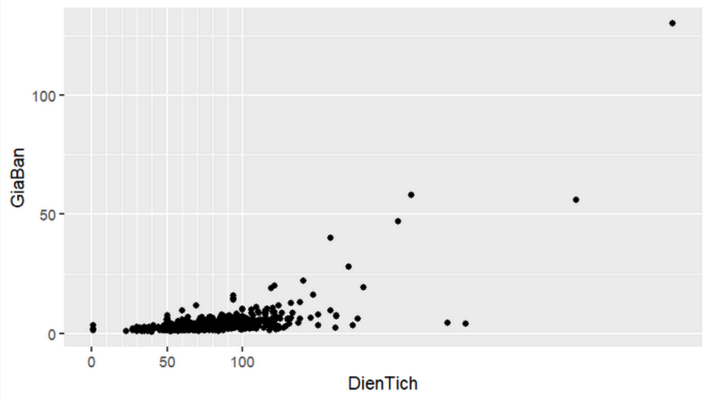
Labels:

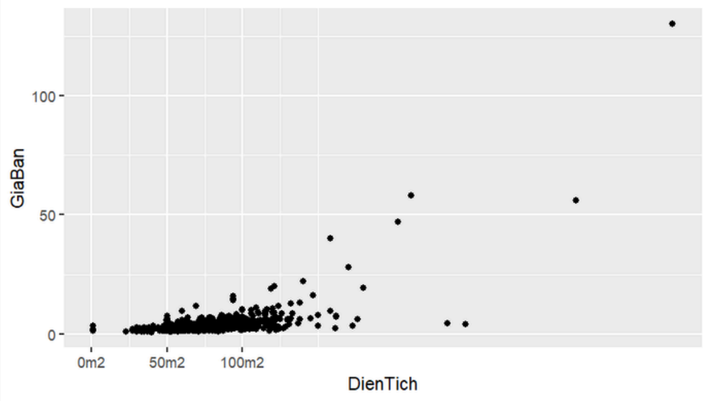
- **Setting Labels**: Assign text to breaks using the labels argument.
- **Removing Labels:** Use labels = NULL to remove axis or legend labels but keep breaks.
- **Handling Overlap:** Even with ample space, axis labels may overlap, requiring careful customization.



base + scale_x_continuous(breaks = c(0, 50, 100))

base + scale_x_continuous(breaks = c(0, 50, 100), minor_breaks = seq(0, 100, 10))





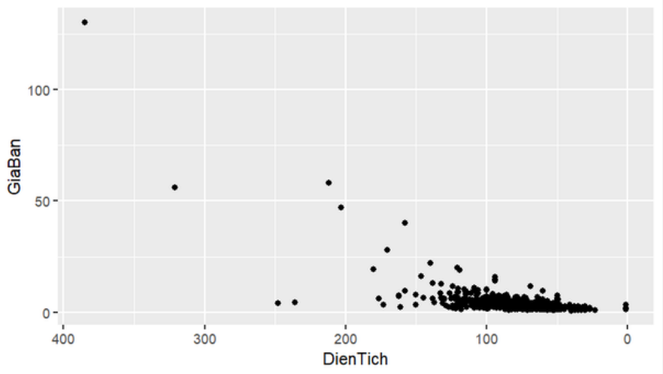
base + scale_x_continuous(breaks = c(0, 50, 100), labels = c("0m2", "50m2", "100m2"))

POSITION SCALES AND AXES | TRANSFORMATION

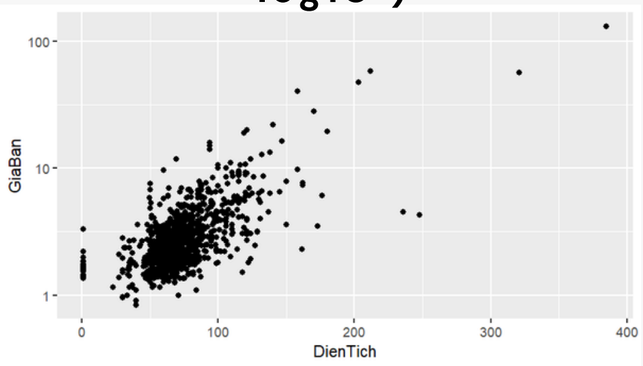
- **Purpose:** Modify how data maps to aesthetics for continuous data.
- **Transformer Argument**: Use trans for various transformations.
- **Common Transformers:** asn, exp, log, log10, log2, logit, ...
- **Timing:** Transformations occur before statistical summaries. Use coord_trans() to transform after.

```
base <- ggplot(data = house, aes(DienTich, GiaBan)) + geom_point()
```

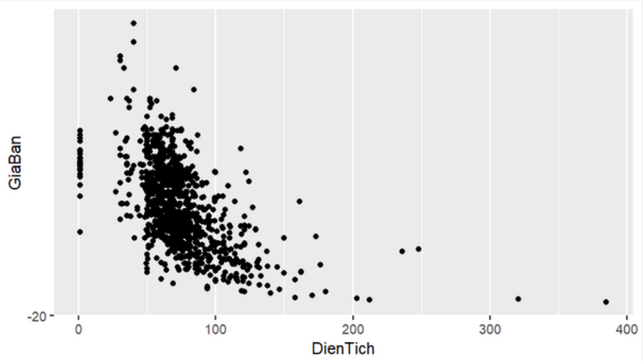
```
base +
scale_x_reverse()
```




```
base +
scale_y_continuous(trans =
"log10")
```




```
base + scale_y_continuous(trans =
scales::reciprocal_trans()
```



POSITION SCALES AND AXES | DATE-TIME

Date-time Position Scales:

- **Handling Dates:** Use scale_x_date() for dates and scale_x_datetime() for date-times. Convert formats with as.Date(), as.POSIXct(), or hms::as_hms().

```
the_year <- as.Date(c("2023-01-01", "2023-12-31"))
set_breaks <- scales::breaks_width("1 month")
set_breaks(the_year)
```

```
[1] "2023-01-01" "2023-02-01" "2023-03-
01" "2023-04-01" "2023-05-01"
 [6] "2023-06-01" "2023-07-01" "2023-08-
01" "2023-09-01" "2023-10-01"
[11] "2023-11-01" "2023-12-01" "2024-01-01"
```

```
set_breaks <- scales::breaks_width("1
month", offset = 10)
set_breaks(the_year)
```

```
[1] "2023-01-11" "2023-02-11" "2023-03-11"
"2023-04-11" "2023-05-11"
 [6] "2023-06-11" "2023-07-11" "2023-08-11"
"2023-09-11" "2023-10-11"
[11] "2023-11-11" "2023-12-11" "2024-01-11"
```

POSITION SCALES AND AXES | DATE-TIME

Breaks:

- **Precise Control:** Use date_breaks to set breaks by units (e.g., "2 weeks", "15 years").
- **Offsets:** Align breaks with specific events using offsets.

Minor Breaks:

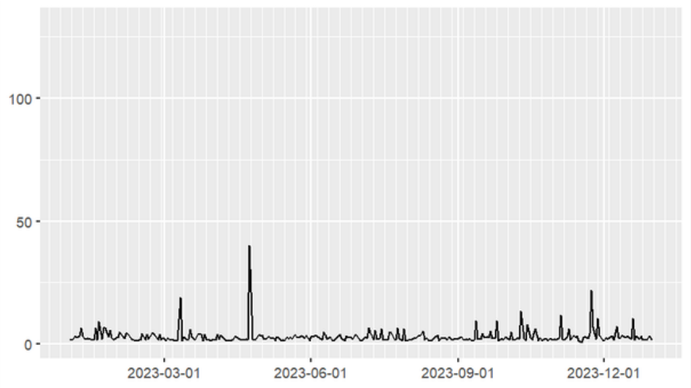
- **Additional Detail:** Use date_minor_breaks for finer grid lines between major breaks.
- **Visual Prominence:** Adjust theme settings to highlight grid lines.

Labels:

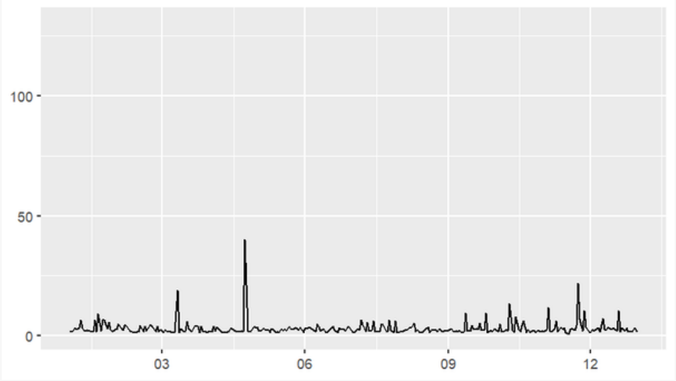
- **Format:** Use date_labels with format strings (e.g., "%d/%m/%Y").
- **Automatic:** Use scales::label_date() and label_date_short().
- **Overlap:** Add line breaks (e.g., "%B\n%Y") for clarity.

```
base <- ggplot(data, aes(x = NgayCapNhat, y = GiaBan)) + geom_line(na.rm = TRUE) + labs(x = NULL, y = NULL)
lim <- as.Date(c("2023-01-01", "2023-06-31"))
```

```
base + scale_x_date(limits = lim,
date_breaks = "3 month",
date_minor_breaks = "1 week")
```



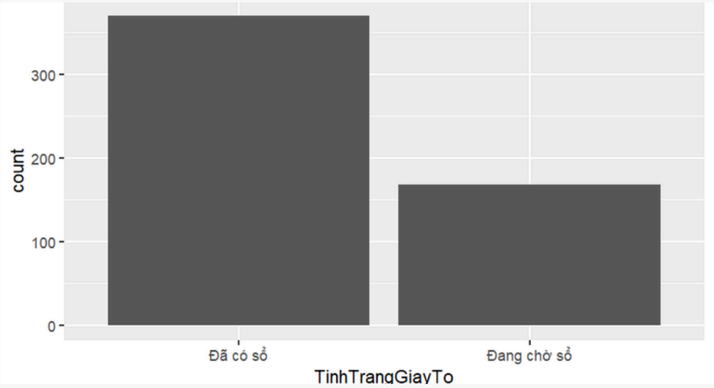

```
base + scale_x_date(limits =
lim, date_breaks = "3 month",
date_labels = "%m")
```



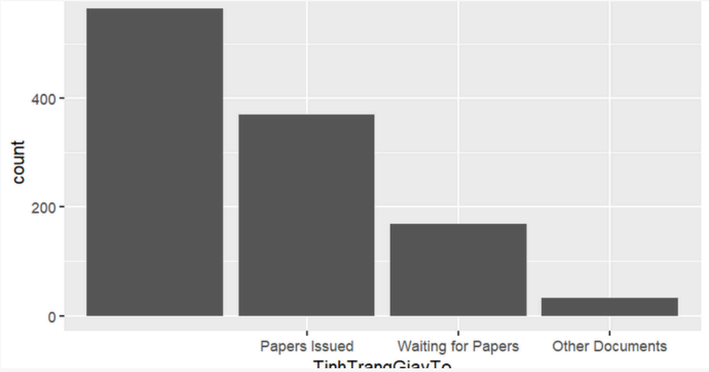
POSITION SCALES AND AXES | DISCRETE

```
base <- ggplot(data, aes(x = TinhTrangGiayTo)) + geom_bar()
```

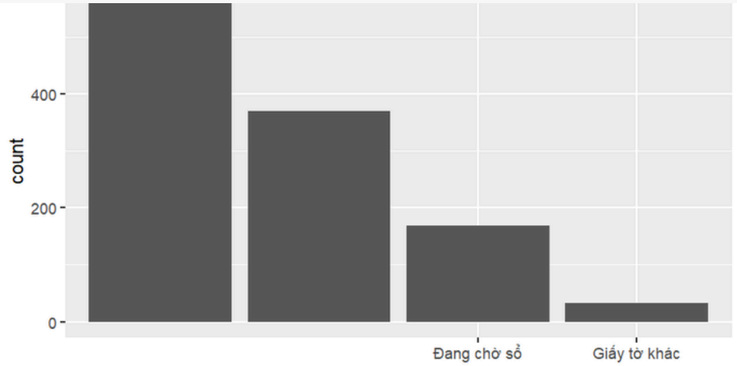
```
base +
scale_x_discrete(limits =
c("Đã có sổ", "Đang chờ sổ"))
```




```
base + scale_x_discrete(breaks
= c("Giấy tờ khác", "Đang chờ
sổ", "Đã có sổ"), labels = c("Other
Documents", "Waiting for
Papers", "Papers Issued"))
```



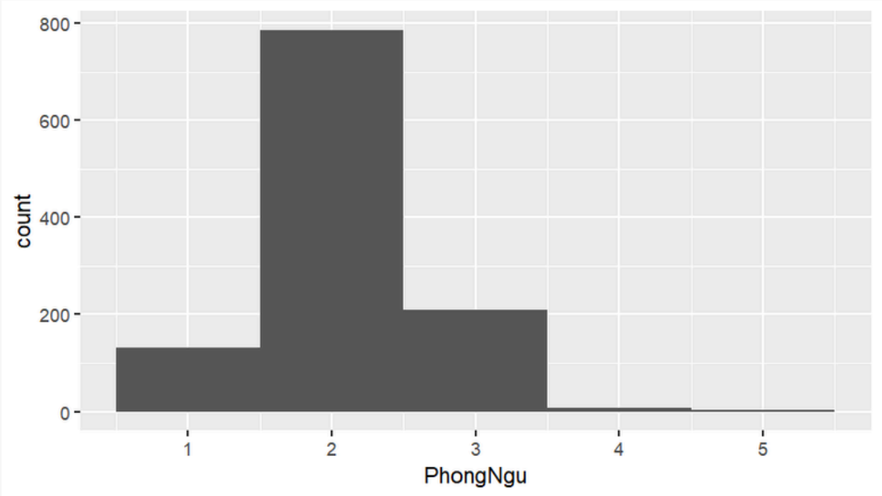

```
base + scale_x_discrete(breaks =
c("Giấy tờ khác", "Đang chờ sổ"))
```



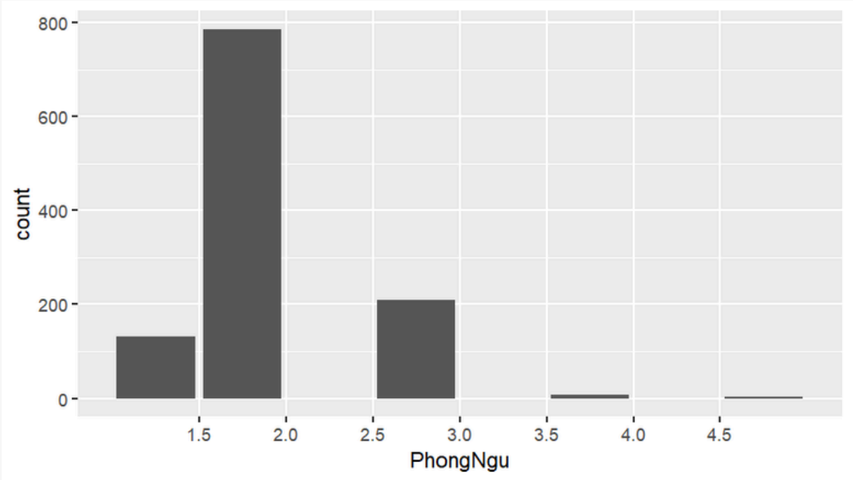
POSITION SCALES AND AXES | BINNED

```
base <- ggplot(data, aes(PhongNgu))
```

```
base + geom_histogram(bins = 5)
```




```
base + geom_bar() +
scale_x_binned()
```



COLOUR SCALES AND LEGENDS

colorBlindness::displayAllColors(rainbow(6))

- **Color Theory**: Understand RGB and HCL color spaces for better color mapping.
- **Color Blindness:** Use tools like the dichromat and colorBlindness packages to ensure accessibility.
- **HCL Color Space:**
  - Hue: Measured in degrees, represents the color name.
  - Chroma: Indicates color purity, influenced by luminance.
  - Luminance: Represents brightness, from 0 (black) to 1 (white).

COLOUR SCALES AND LEGENDS

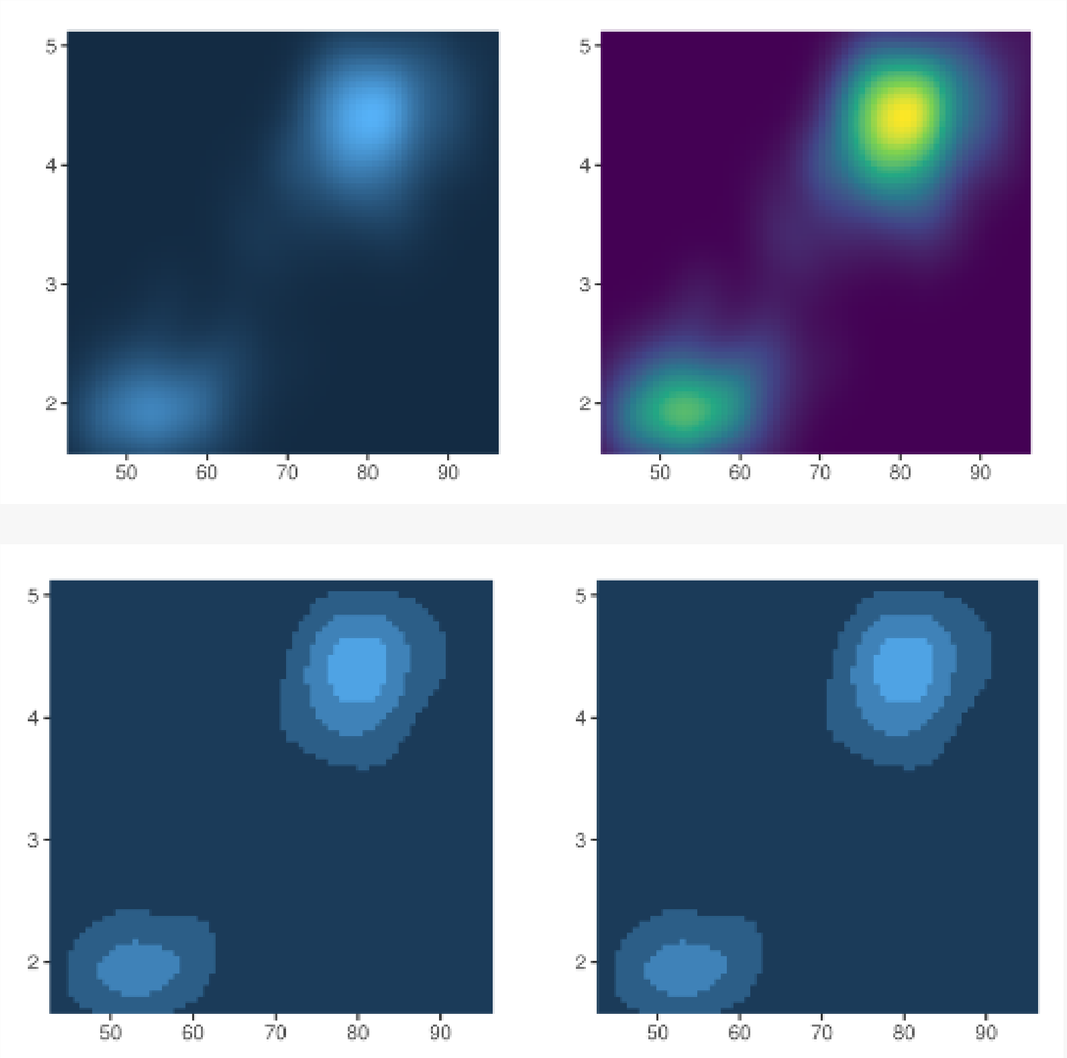
- **Color Theory**: Understand RGB and HCL color spaces for better color mapping.
- **Color Blindness:** Use tools like the dichromat and colorBlindness packages to ensure accessibility.
- **HCL Color Space:**
  - Hue: Measured in degrees, represents the color name.
  - Chroma: Indicates color purity, influenced by luminance.
  - Luminance: Represents brightness, from 0 to 1

colorBlindness::displayAllColors(rainbow(6))

colorBlindness::displayAllColors(viridis::viridis(6))

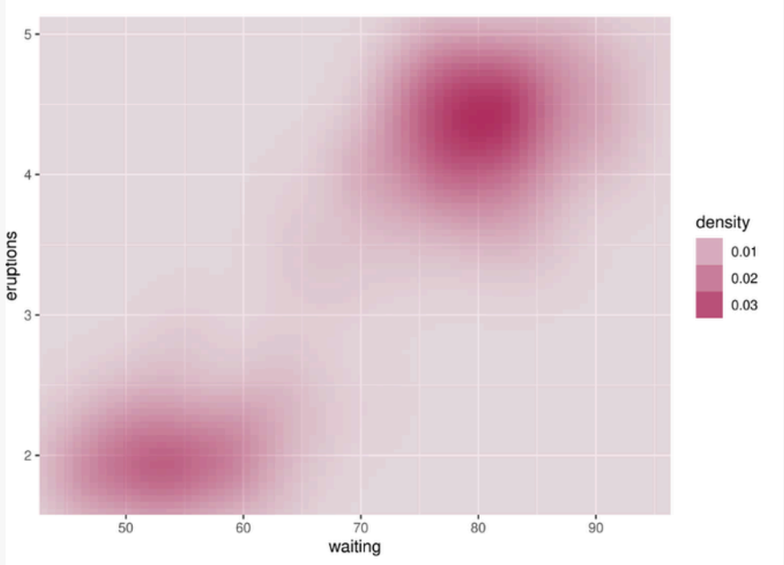
COLOUR SCALES AND LEGENDS

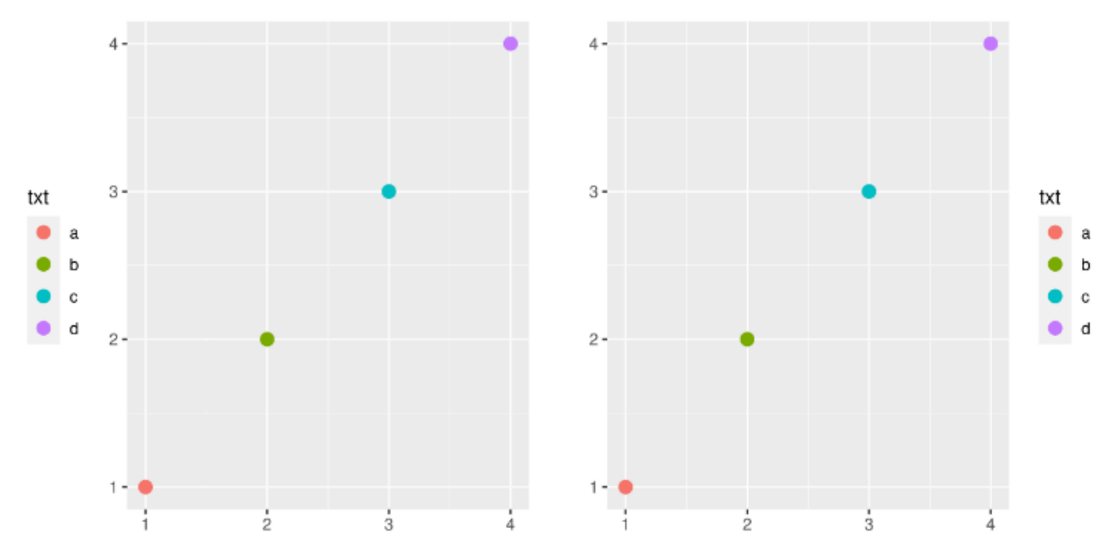
- **Continuous Scales:**
  - Viridis: Provides smooth and perceptually uniform color transitions, often used for gradients from green to yellow.
  - Gradient: Allows creation of custom color blends with scale_fill_gradient and scale_fill_gradient2.
- **Discrete Scales:**
  - Brewer: Offers predefined palettes ideal for categorical data with scale_fill_brewer.
  - Hue & Gray: Vary hues while maintaining consistent lightness with scale_fill_hue and scale_fill_grey.
  - Paletteer: Access additional pre-built palettes from various sources.
- **Binned Scales:**
  - Fixed Colors: scale_fill_fermenter uses fixed color bins without interpolation. Setting too many bins can result in missing colors, triggering a warning.



COLOUR SCALES AND LEGENDS

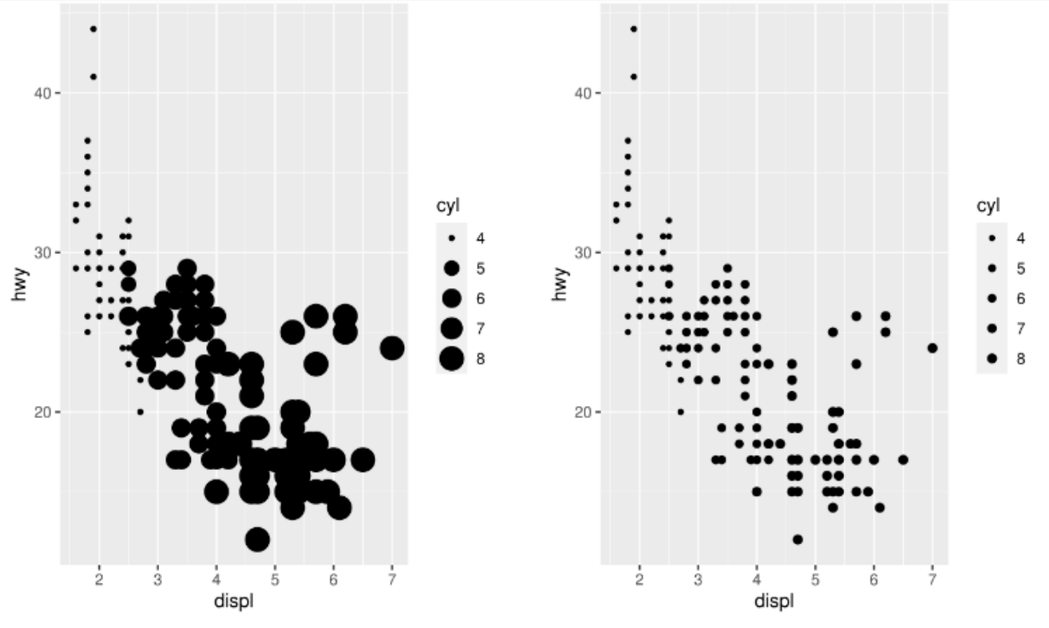
- **Alpha Scales:**
  - Purpose: Map transparency to data values, useful for down-weighting less important observations.
  - Usage: scale_alpha() for continuous alpha scales.
    - Example: ggplot(faithfuld, aes(waiting, eruptions, alpha = density)) + geom_raster(fill = "maroon")
- **Legend Position:**
  - Control with theme(): Position and justify legends using legend.position.
  - Options: “right”, “left”, “top”, “bottom”, “none”
    - Example: theme(legend.position = "left")
  - Custom Positioning: Use numeric vectors for inside plot placement.
    - Example: theme(legend.position = c(0, 1), legend.justification = c(0, 1))

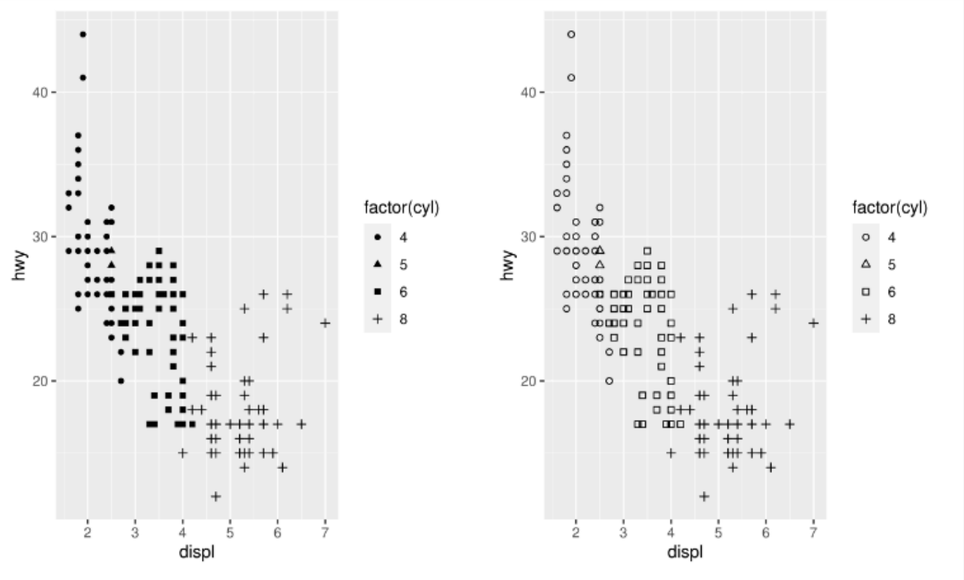




OTHER AESTHETICS

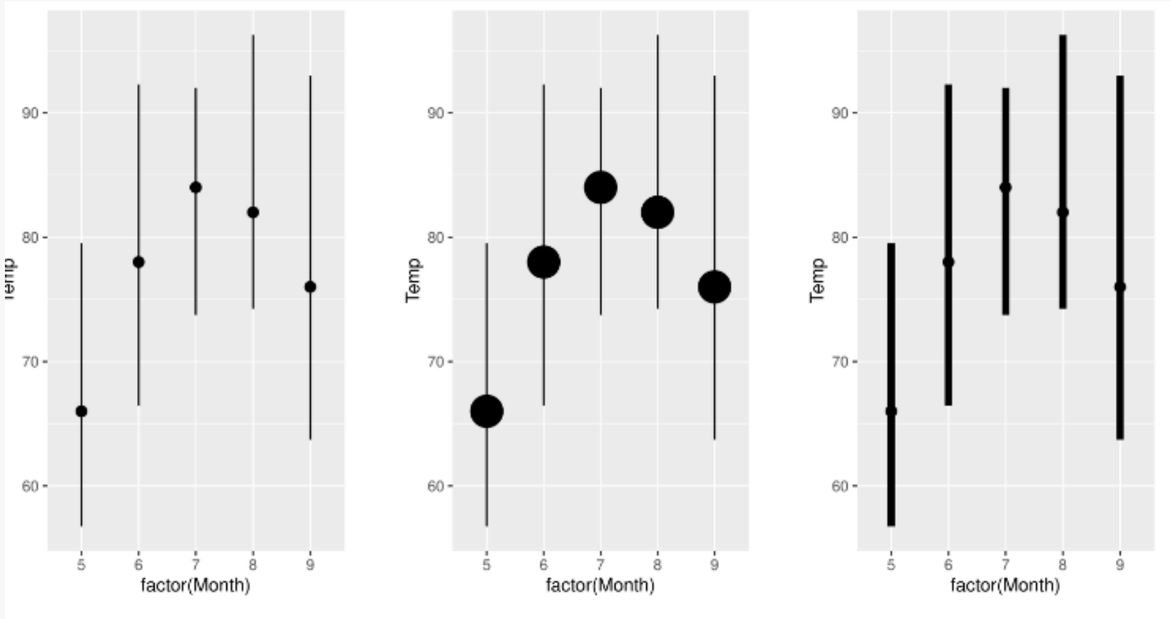
- **Size Scales:**
  - Mapping Data: scale_size() maps data to size, scaling by area for perceptual accuracy.
  - Options: Includes scale_size_area() for zero mapping, scale_radius() for radius-based scaling, and scale_size_binned() for discrete size categories with custom legends via guide_bins().
- **Shape Scales:**
  - Mapping Categories: scale_shape() maps data to shape, suitable for up to 6 categories.
  - Customization: Use scale_shape_manual() to specify marker types for each data value.

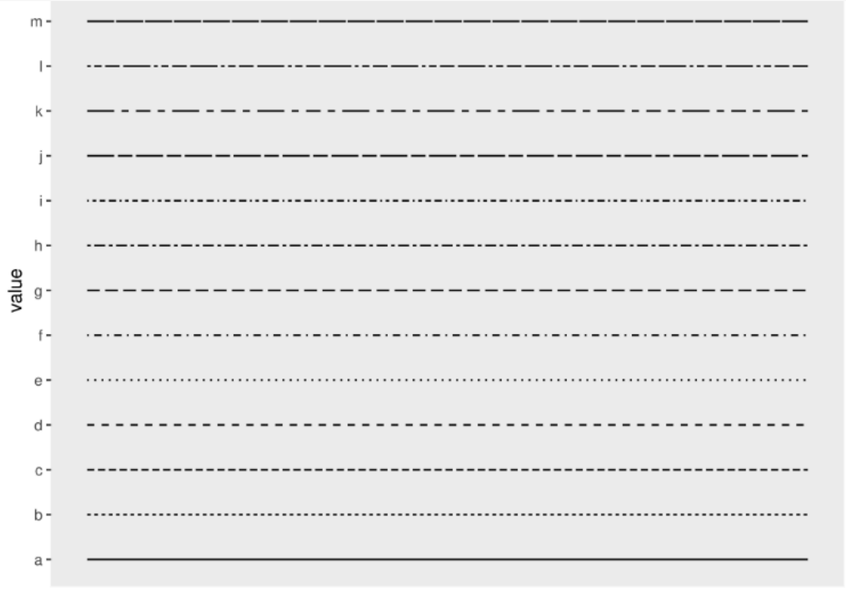




OTHER AESTHETICS

- **Line Width:**
  - Control: linewidth aesthetic controls line width.
  - Default: scale_linewidth() increases width linearly with data values.
  - Usage: Example: geom_line(aes(linewidth = Month)) + scale_linewidth(range = c(0.5, 3))
  - Binned: Use scale_linewidth_binned() for discrete categories.
- **Line Type:**
  - Mapping: linetype aesthetic maps variables to line types, ideal for few categories.
  - Default: scale_linetype() uses 13 predefined types.
  - Custom: Use scale_linetype_manual() or custom functions for specific line patterns.
  - Limit: Best for up to 5 categories; more may reduce readability.

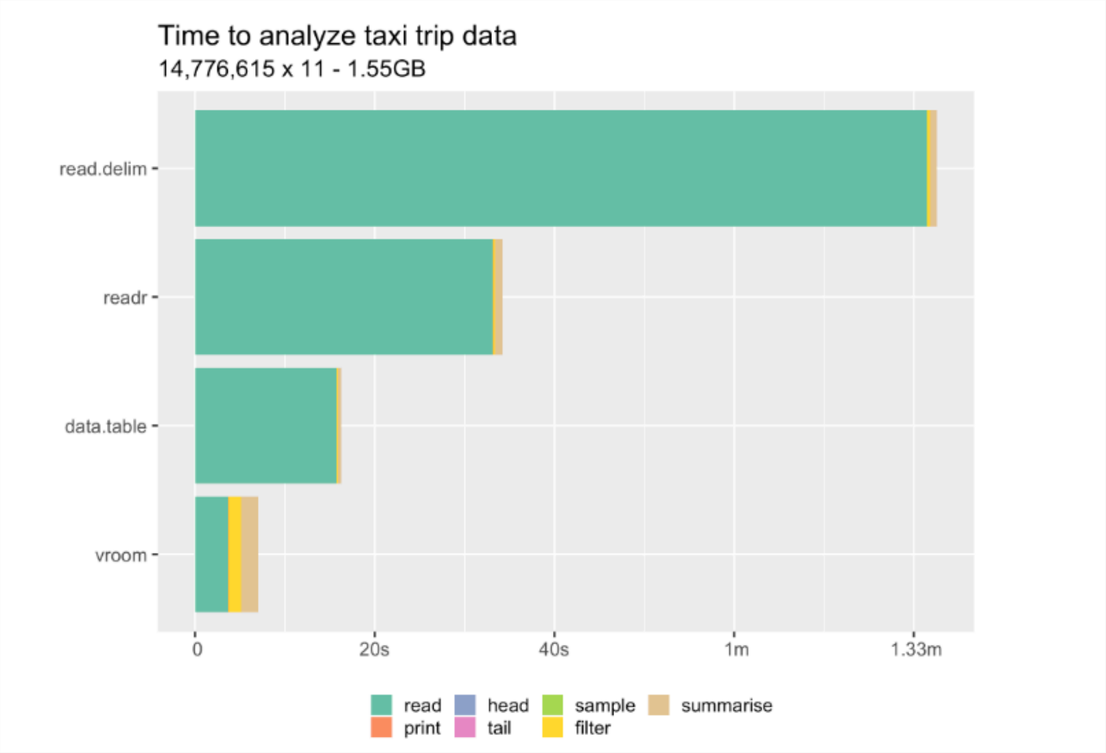




Q&A

Question 01 | How do we keep up with growth of data when using the tool?

- **Data Manipulation**: Using dplyr for data wrangling tasks like filtering, summarizing, and joining datasets.
- **Big Data Handling**: Strategies for managing and analyzing large datasets using data.table and sparklyr
- **Improve your velocity using VROOM package**: According to the VROOM developer, this package reads data faster than others do, because it reads files lazily (it indexes the data and only reads it when is necessary)



Question 02 | How long will we receive the response if the input is large enough or is the response time good enough if we have a large input?

In RStudio, the response time for processing large inputs depends on:

1. **Computational Power:** Your system's CPU and memory capabilities.
2. **Code Optimization:** Efficient code, such as using vectorized operations and avoiding loops.
3. **Data Size:** Larger and more complex data take longer to process.
4. **Packages and Libraries**: Some are optimized for speed and handle large datasets better.
5. **Parallel Processing:** Distributing tasks across multiple cores reduces computation time.

```
> large_data <- `IMDb.Movies.India`
> start_time <- Sys.time()
> filtered_data <- large_data[large_data$Year == 2020, ]
> end_time <- Sys.time()
> time_taken <- as.numeric(difftime(end_time, start_time, units = "secs")) * 1000
> print(paste("Time taken: ", time_taken, "ms"))
[1] "Time taken:  952.755928039551 ms"
```

```
> source("~/.active-rstudio-document")
[1] "Time taken:  5.33914566040039 ms"
> source("~/.active-rstudio-document")
[1] "Time taken:  0.158786773681641 ms"
```

THANK YOU