Plotting with ggplot2 in R

Content

- Produce scatter plots, boxplots, and time series plots using ggplot.
- Set universal plot settings.
- Describe what faceting is and apply faceting in ggplot.
- Modify the aesthetics of an existing ggplot plot (including axis labels and colour).
- Build complex and customized plots from data in a data frame.

Get the data

- Download package with our data in

```
devtools::install_github("n8thangreen/dataPakistan")
```

- Load the package with the data in

```
library(dataPakistan)
```

- Take a look at what data is available

```
system.file("extdata", package = "dataPakistan")
```

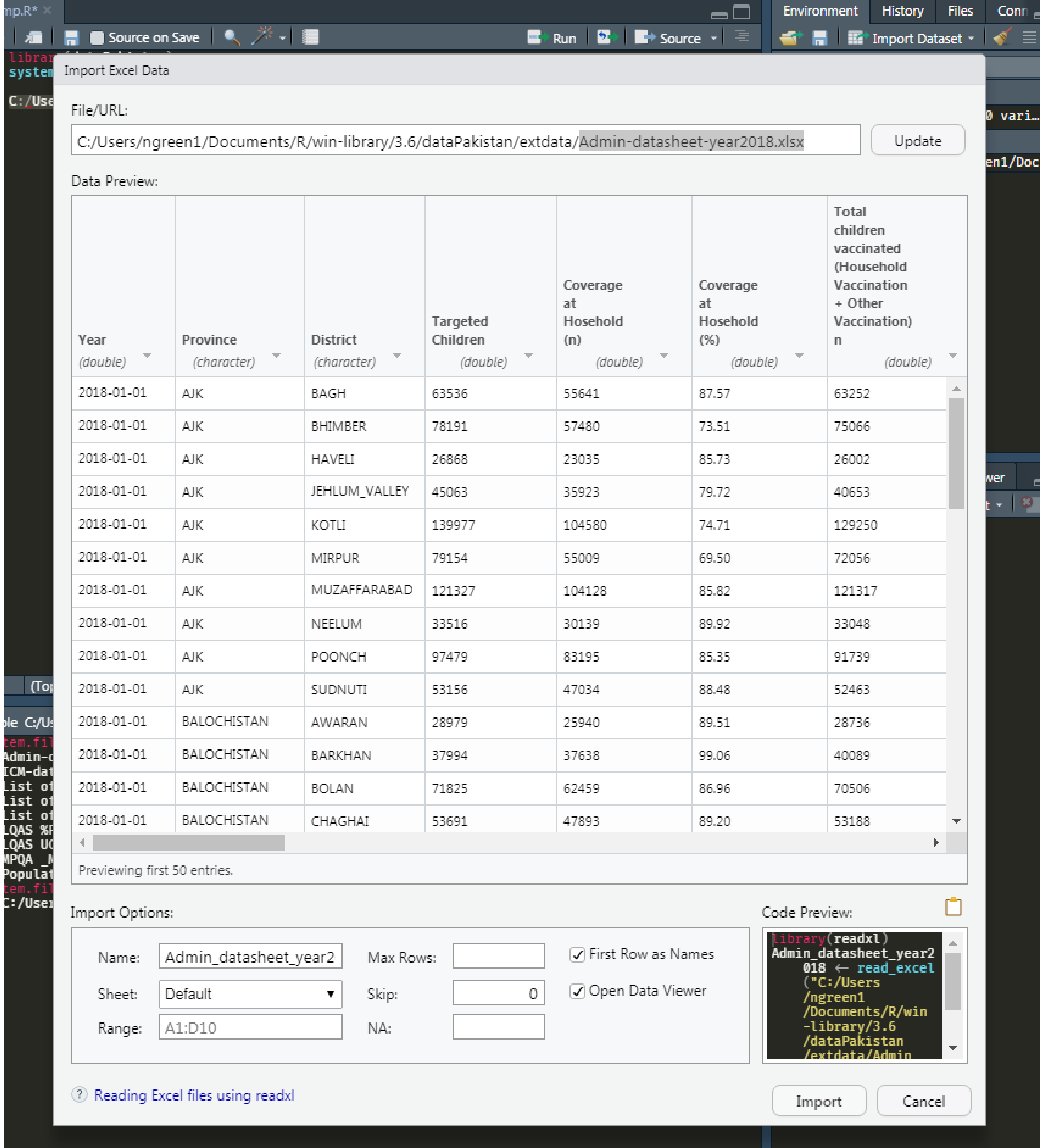
Import data

- Using one of the ways already mentioned e.g.
- Use Environment|Import Dataset pane
- Import

Admin-datasheet-year2018.xlsx

- Or type in the console

```
> library(readxl)
> dat <- read_excel("C:/Users/ngreen1/Documents/R/win-
library/3.6/dataPakistan/extdata/Admin-datasheet-
year2018.xlsx")
> View(dat)
```



Import Excel Data

File/URL:

C:/Users/ngreen1/Documents/R/win-library/3.6/dataPakistan/extdata/Admin-datasheet-year2018.xlsx

Data Preview:

Year (double)	Province (character)	District (character)	Targeted Children (double)	Coverage at Hosehold (n) (double)	Coverage at Hosehold (%) (double)	Total children vaccinated (Household Vaccination + Other Vaccination) n (double)
2018-01-01	AJK	BAGH	63536	55641	87.57	63252
2018-01-01	AJK	BHIMBER	78191	57480	73.51	75066
2018-01-01	AJK	HAVELI	26868	23035	85.73	26002
2018-01-01	AJK	JEHLUM_VALLEY	45063	35923	79.72	40653
2018-01-01	AJK	KOTLI	139977	104580	74.71	129250
2018-01-01	AJK	MIRPUR	79154	55009	69.50	72056
2018-01-01	AJK	MUZAFFARABAD	121327	104128	85.82	121317
2018-01-01	AJK	NEELUM	33516	30139	89.92	33048
2018-01-01	AJK	POONCH	97479	83195	85.35	91739
2018-01-01	AJK	SUDNUTI	53156	47034	88.48	52463
2018-01-01	BALOCHISTAN	AWARAN	28979	25940	89.51	28736
2018-01-01	BALOCHISTAN	BARKHAN	37994	37638	99.06	40089
2018-01-01	BALOCHISTAN	BOLAN	71825	62459	86.96	70506
2018-01-01	BALOCHISTAN	CHAGHAI	53691	47893	89.20	53188

Previewing first 50 entries.

Import Options:

Name: Admin_datasheet_year2

Max Rows:

☑ First Row as Names

Sheet: Default

Skip: 0

☑ Open Data Viewer

Range: A1:D10

NA:

Code Preview:

```
library(readxl)
Admin_datasheet_year2
    018 <- read_excel
    ("C:/Users
    /ngreen1
    /Documents/R/win
    -library/3.6
    /dataPakistan
    /extdata/Admin
```

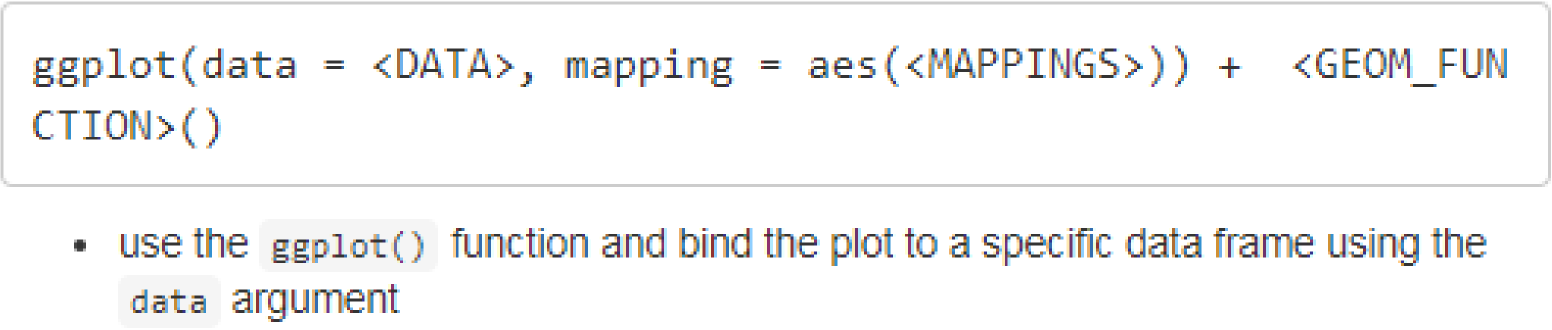
? Reading Excel files using readxl

What is ggplot?

- ggplot2 is a plotting package that makes it simple to create complex plots from data in a data frame
- Provides a more programmatic interface for specifying what variables to plot, how they are displayed, and general visual properties
- Only need minimal changes if the underlying data change or if we decide to change from a bar plot to a scatter plot
- Helps in creating publication quality plots with minimal amounts of adjustments and tweaking.

- ggplot2 functions like data in the ‘long’ format
  - i.e., a column for every dimension, and a row for every observation. Well-structured data will save you lots of time when making figures with ggplot2

- ggplot graphics are built step by step by adding new elements
- Adding layers in this fashion allows for extensive flexibility and customization of plots.
- To build a ggplot, we will use the following basic template that can be used for different types of plots:



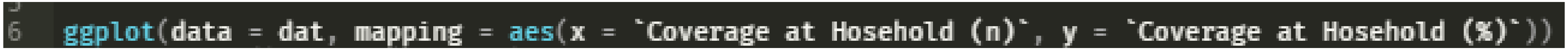
```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +  <GEOM_FUN
CTION>()
```

- use the `ggplot()` function and bind the plot to a specific data frame using the `data` argument

```
ggplot(data = dat,
```

aes

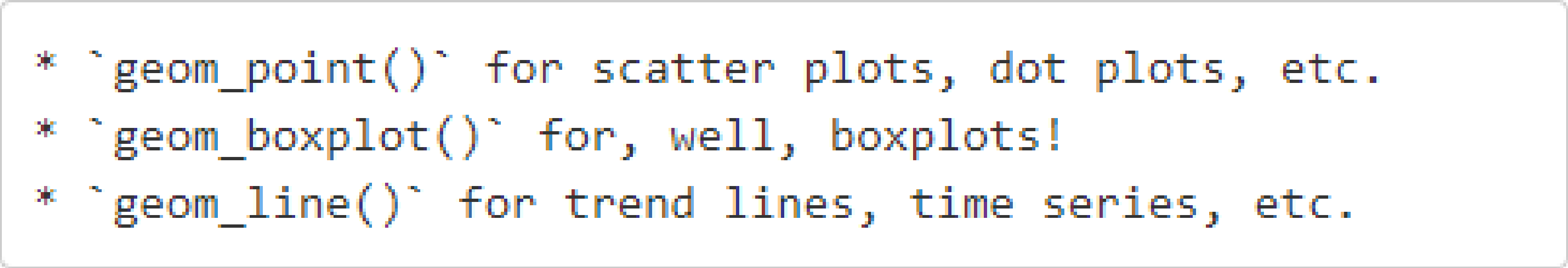
- Define a mapping (using the aesthetic (aes) function), by selecting the variables to be plotted and specifying how to present them in the graph
- e.g. as x/y positions or characteristics such as size, shape, colour, etc.



```r
ggplot(data = dat, mapping = aes(x = `Coverage at Hosehold (n)`, y = `Coverage at Hosehold (%)`))
```

geoms

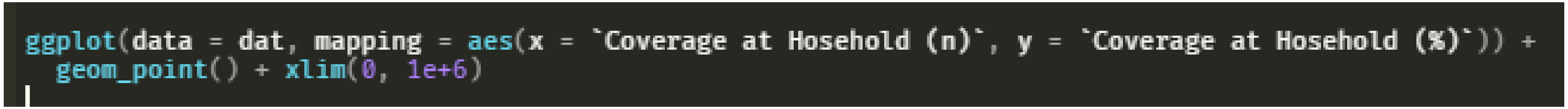
- add ‘geoms’
  - graphical representations of the data in the plot (points, lines, bars)
- ggplot2 offers many different geoms; we will use some common ones today, including:



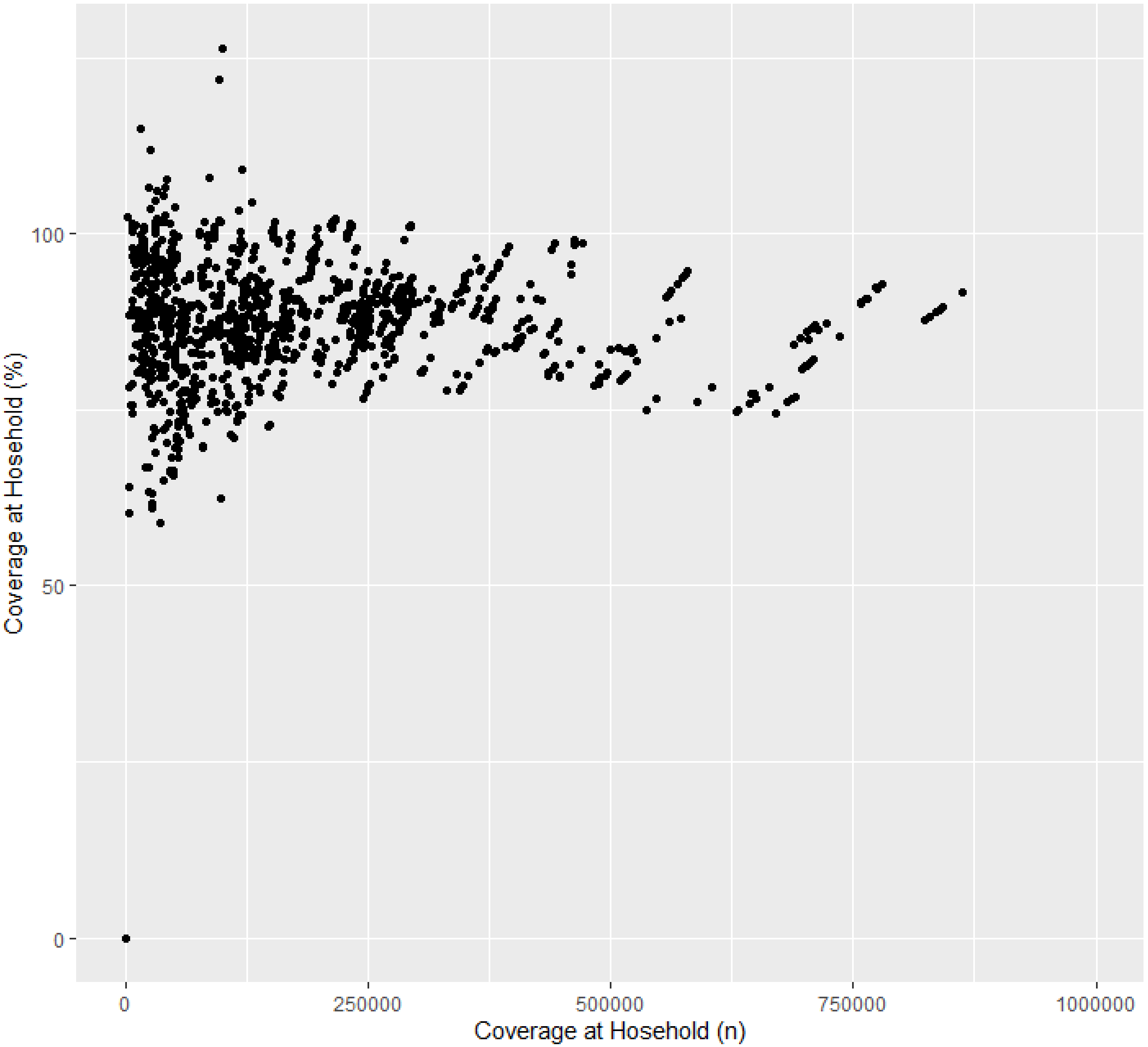
```
* `geom_point()` for scatter plots, dot plots, etc.
* `geom_boxplot()` for, well, boxplots!
* `geom_line()` for trend lines, time series, etc.
```

+ operator

- To add a geom to the plot use the + operator
- Because we have two continuous variables, let’s use geom_point() first:



```
ggplot(data = dat, mapping = aes(x = `Coverage at Hosehold (n)`, y = `Coverage at Hosehold (%)`)) +
  geom_point() + xlim(0, 1e+6)
```

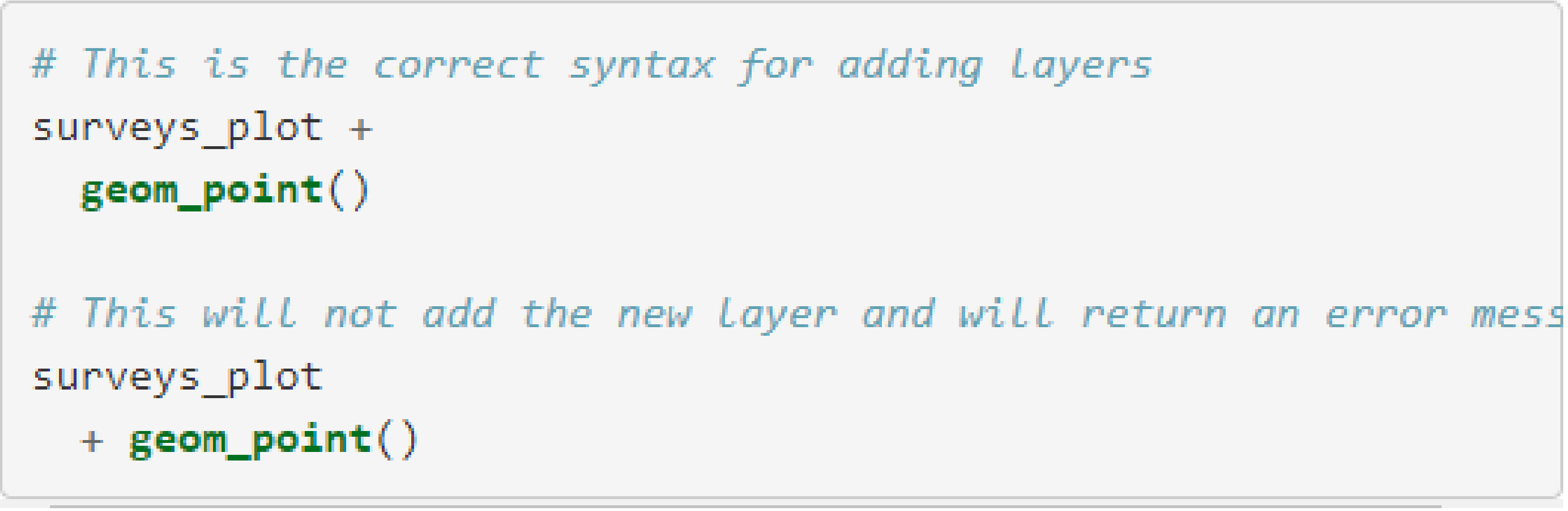

Coverage at Hosehold (%)
100
50
0
0
250000
500000
750000
1000000
Coverage at Hosehold (n)

- The + in the ggplot2 package is particularly useful because it allows you to modify existing ggplot objects.
- This means you can easily set up plot templates and conveniently explore different types of plots, so the above plot can also be generated with code like this:

```
my_plot <- ggplot(data = dat, mapping = aes(x = `Coverage at Hosehold (n)`, y = `Coverage at Hosehold (%)`))


my_plot + geom_point() + xlim(0, 1e+6)
```

Notes

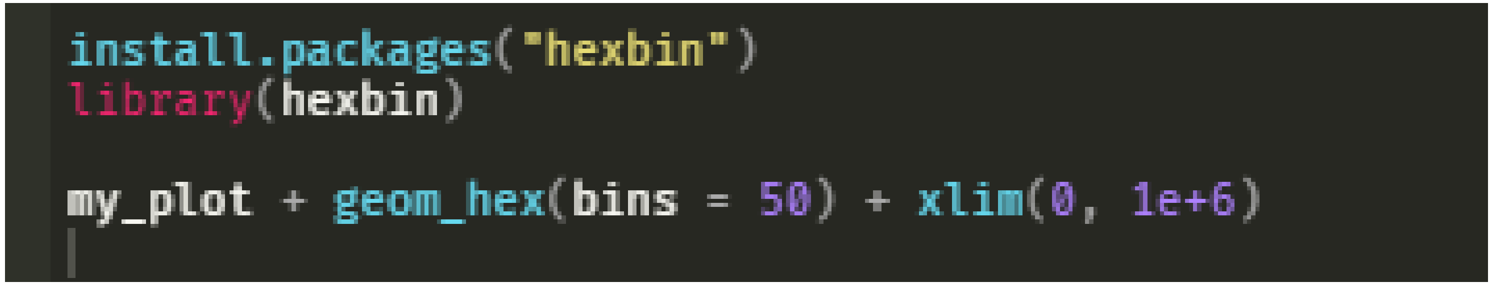
- Anything you put in the ggplot() function can be seen by any geom layers that you add (i.e., these are universal plot settings). This includes the x- and y-axis mapping you set up in aes().
- You can also specify mappings for a given geom independently of the mappings defined globally in the ggplot() function.
- The + sign used to add new layers must be placed at the end of the line containing the previous layer. If, instead, the + sign is added at the beginning of the line containing the new layer, ggplot2 will not add the new layer and will return an error message.



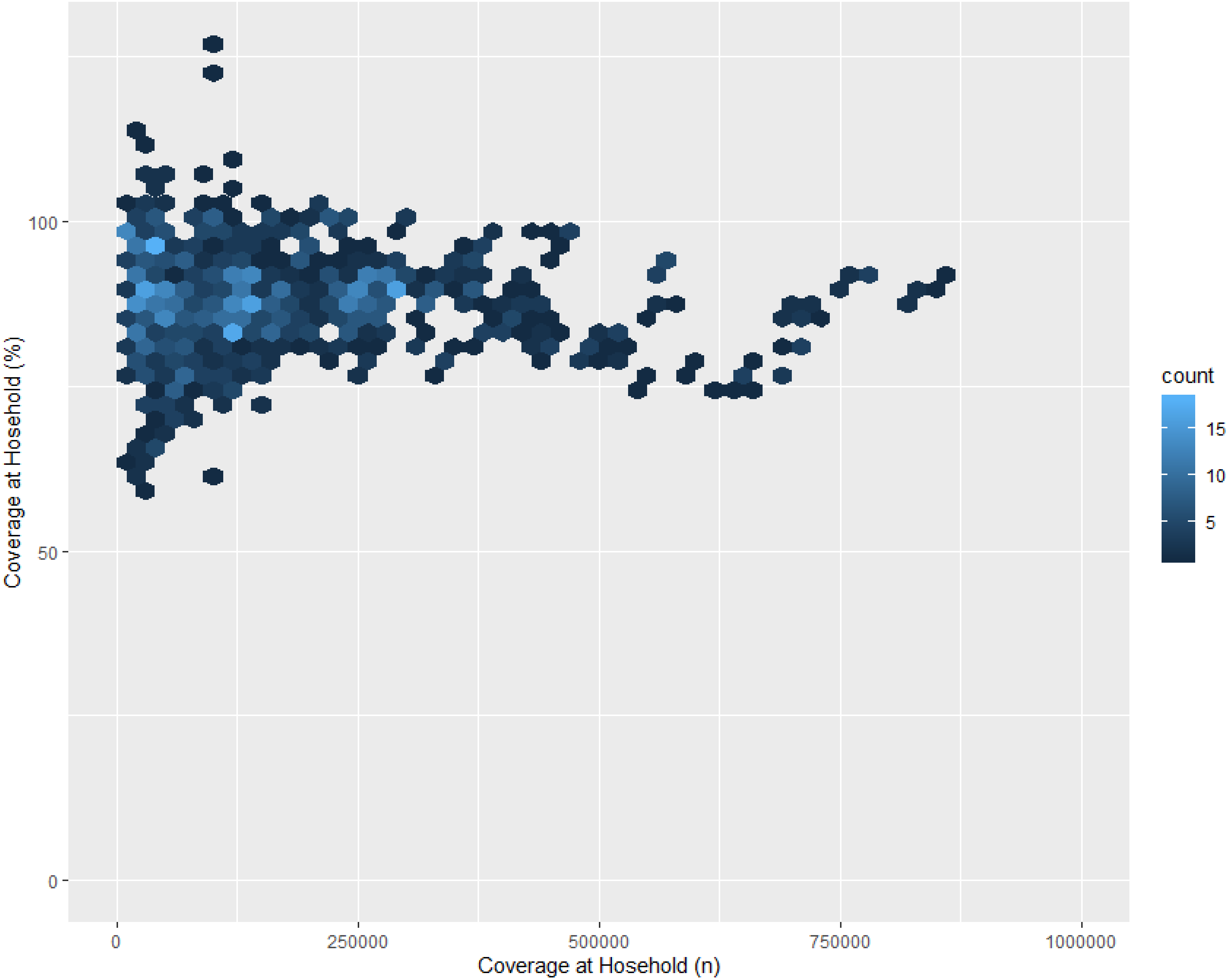
```
# This is the correct syntax for adding layers
surveys_plot +
  geom_point()

# This will not add the new layer and will return an error mess
surveys_plot
  + geom_point()
```

hexbin

- Scatter plots can be useful exploratory tools for small datasets. For data sets with large numbers of observations,overplotting of points can be a limitation of scatter plots. One strategy for handling such settings is to use hexagonal binning of observations. The plot space is tessellated into hexagons. Each hexagon is assigned a colour based on the number of observations that fall within its boundaries. To use hexagonal binning with ggplot2, first install the R package hexbin from CRAN:

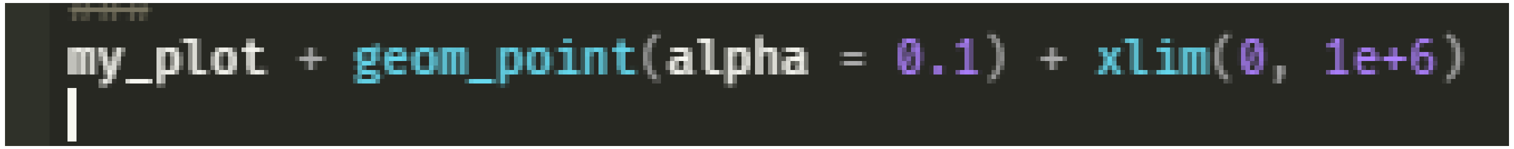


```
install.packages("hexbin")
library(hexbin)

my_plot + geom_hex(bins = 50) + xlim(0, 1e+6)
```

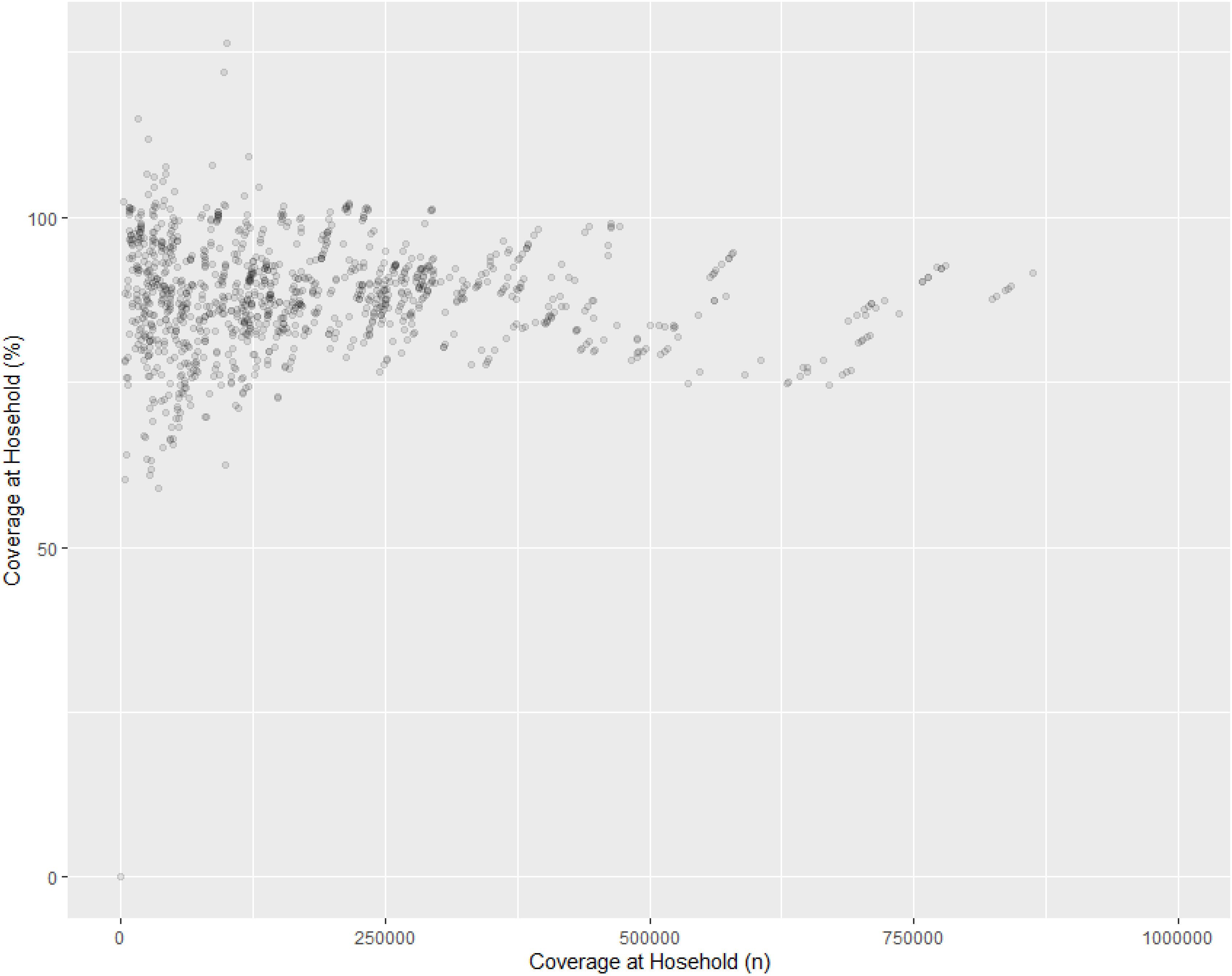

Coverage at Hosehold (%)
100
50
0
Coverage at Hosehold (n)
0
250000
500000
750000
1000000
count
15
10
5

Building plots iteratively

- Building plots with ggplot2 is typically an iterative process. We start by defining the dataset we'll use, lay out the axes, and choose a geom.
- Then, we start modifying this plot to extract more information from it. For instance, we can add transparency (alpha) to avoid overplotting:

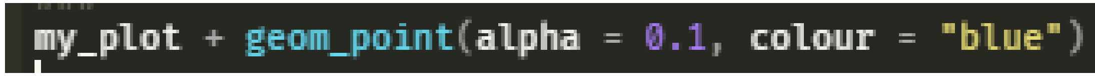


```
####
my_plot + geom_point(alpha = 0.1) + xlim(0, 1e+6)
```

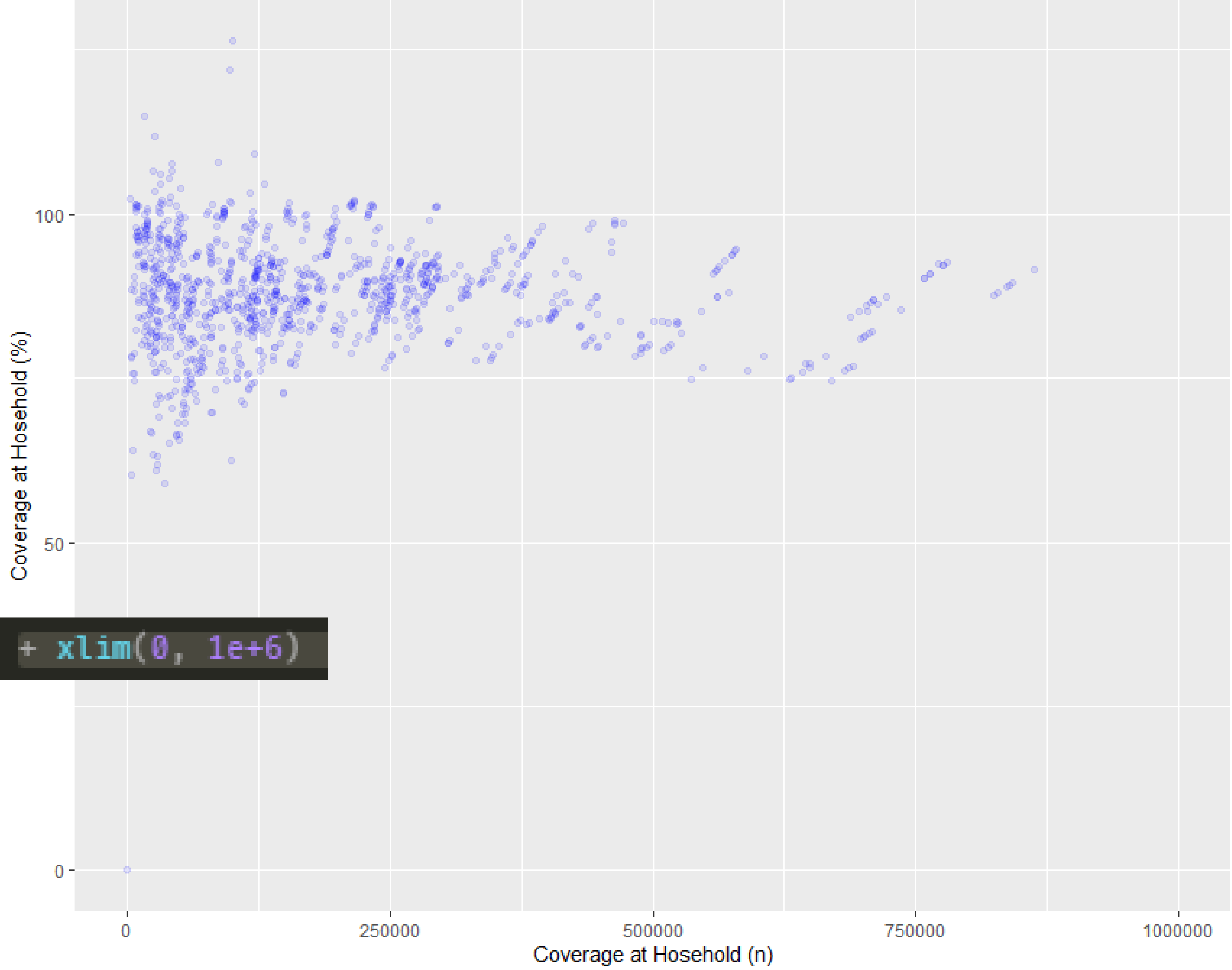

Coverage at Hosehold (%)
100
50
0
0
250000
500000
750000
1000000
Coverage at Hosehold (n)

Colours

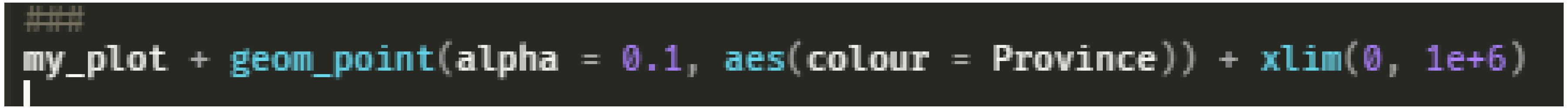
- We can also add colours for all the points:



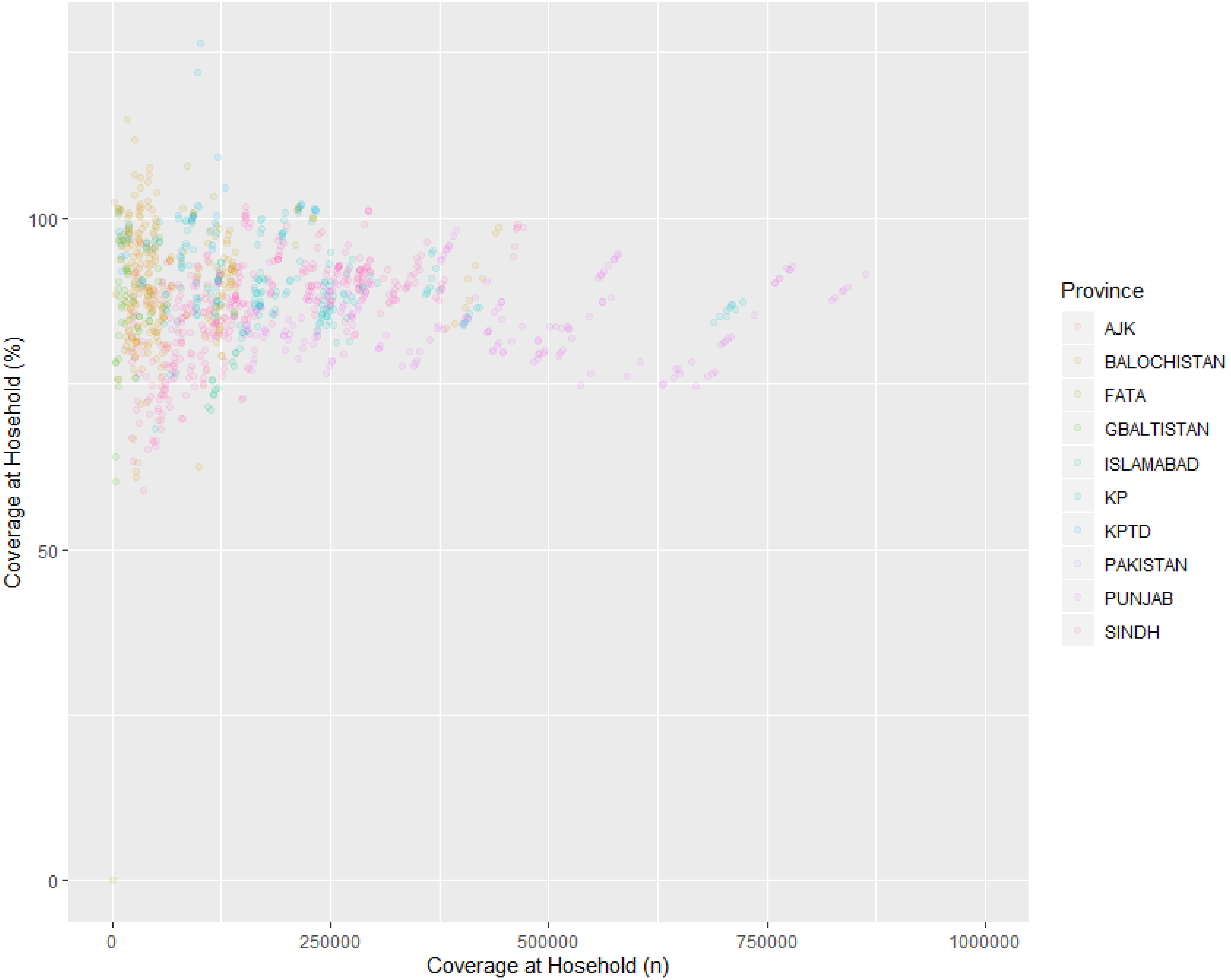
```
my_plot + geom_point(alpha = 0.1, colour = "blue") + xlim(0, 1e+6)
```



- Or to colour each species in the plot differently, you could use a vector as an input to the argument colour.
- ggplot2 will provide a different colour corresponding to different values in the vector.
- Here is an example where we colour with Province:



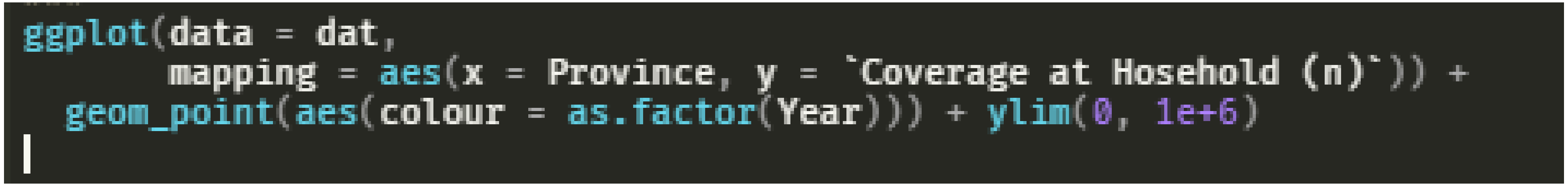
```
####
my_plot + geom_point(alpha = 0.1, aes(colour = Province)) + xlim(0, 1e+6)
```


Coverage at Hosehold (%)
0
50
100
Coverage at Hosehold (n)
0
250000
500000
750000
1000000
Province
AJK
BALOCHISTAN
FATA
GBALTISTAN
ISLAMABAD
KP
KPTD
PAKISTAN
PUNJAB
SINDH

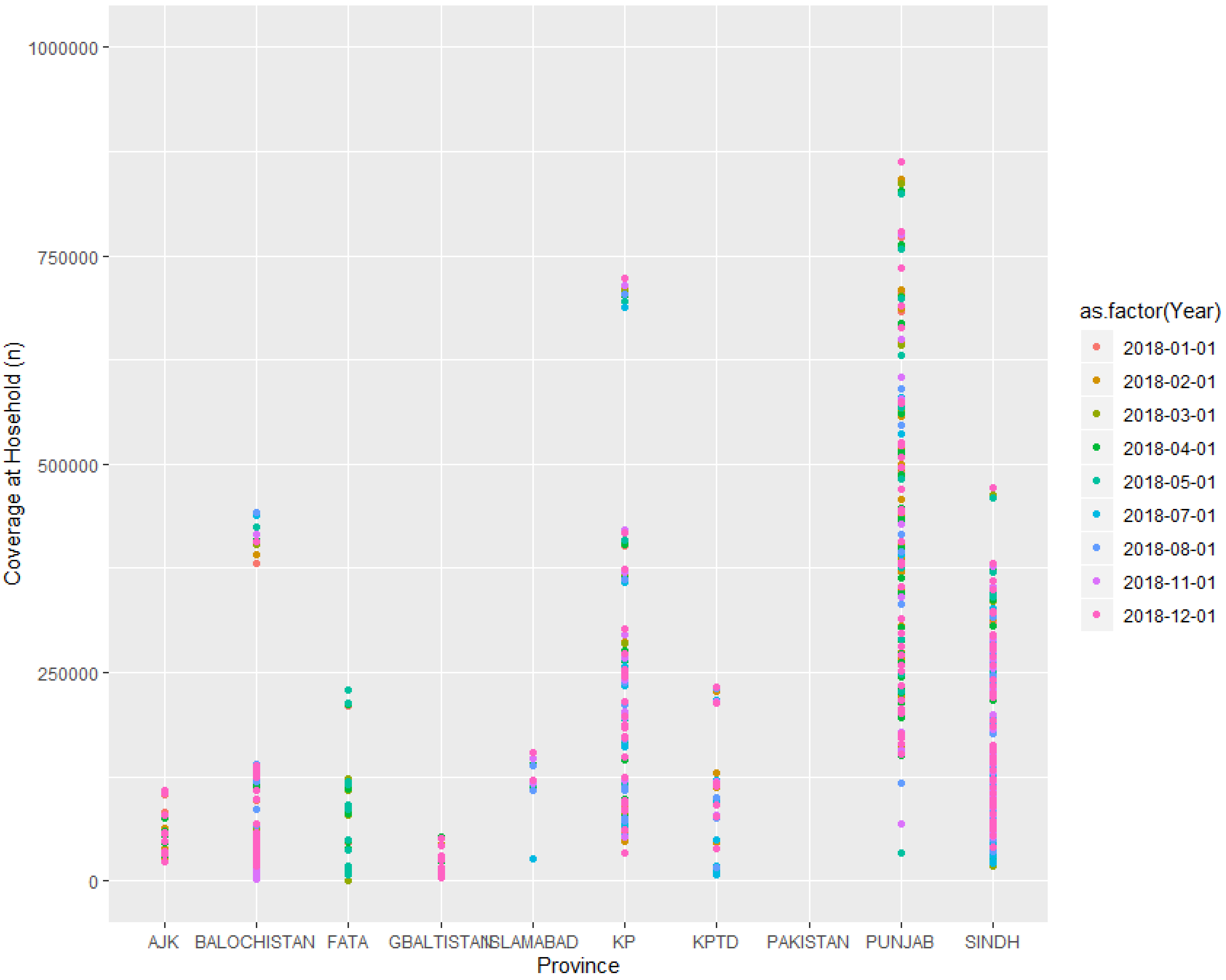
- We can also specify the colors directly inside the mapping provided in the ggplot() function.
- This will be seen by any geom layers and the mapping will be determined by the x- and y-axis set up in aes().

Your turn

- Use what you just learned to create a scatter plot of Coverage at household (n) by Province with the Year showing in different colours.
- Is this a good way to show this type of data?

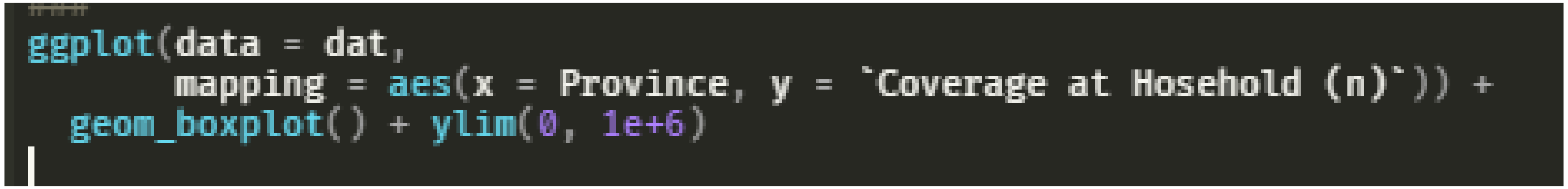


```
ggplot(data = dat,
       mapping = aes(x = Province, y = `Coverage at Hosehold (n)`)) +
  geom_point(aes(colour = as.factor(Year))) + ylim(0, 1e+6)
```

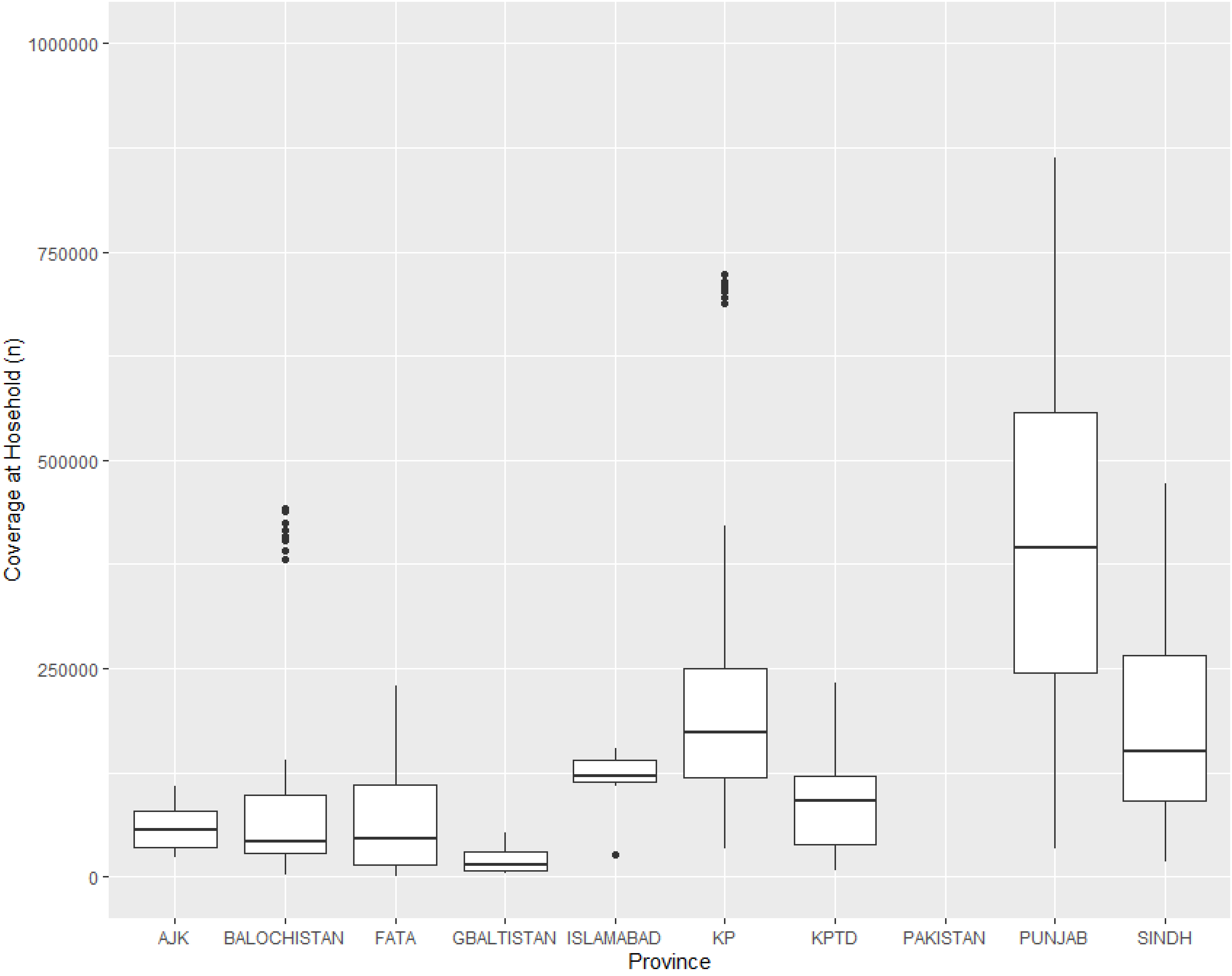

Coverage at Hosehold (n)
1000000
750000
500000
250000
0
AJK
BALOCHISTAN
FATA
GBALTISTAN
ISLAMABAD
KP
KPTD
PAKISTAN
PUNJAB
SINDH
Province
as.factor(Year)
● 2018-01-01
● 2018-02-01
● 2018-03-01
● 2018-04-01
● 2018-05-01
● 2018-07-01
● 2018-08-01
● 2018-11-01
● 2018-12-01

Boxplot

- Visualise distribution with geom_boxplot()

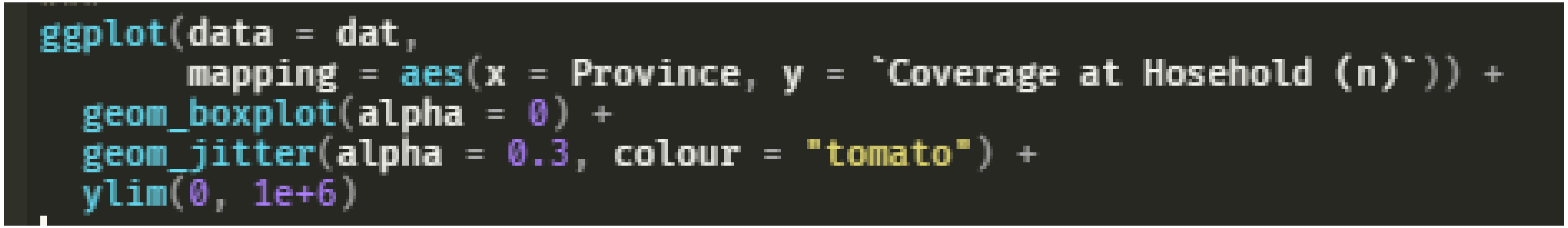


```
####
ggplot(data = dat,
       mapping = aes(x = Province, y = `Coverage at Hosehold (n)`)) +
  geom_boxplot() + ylim(0, 1e+6)
```

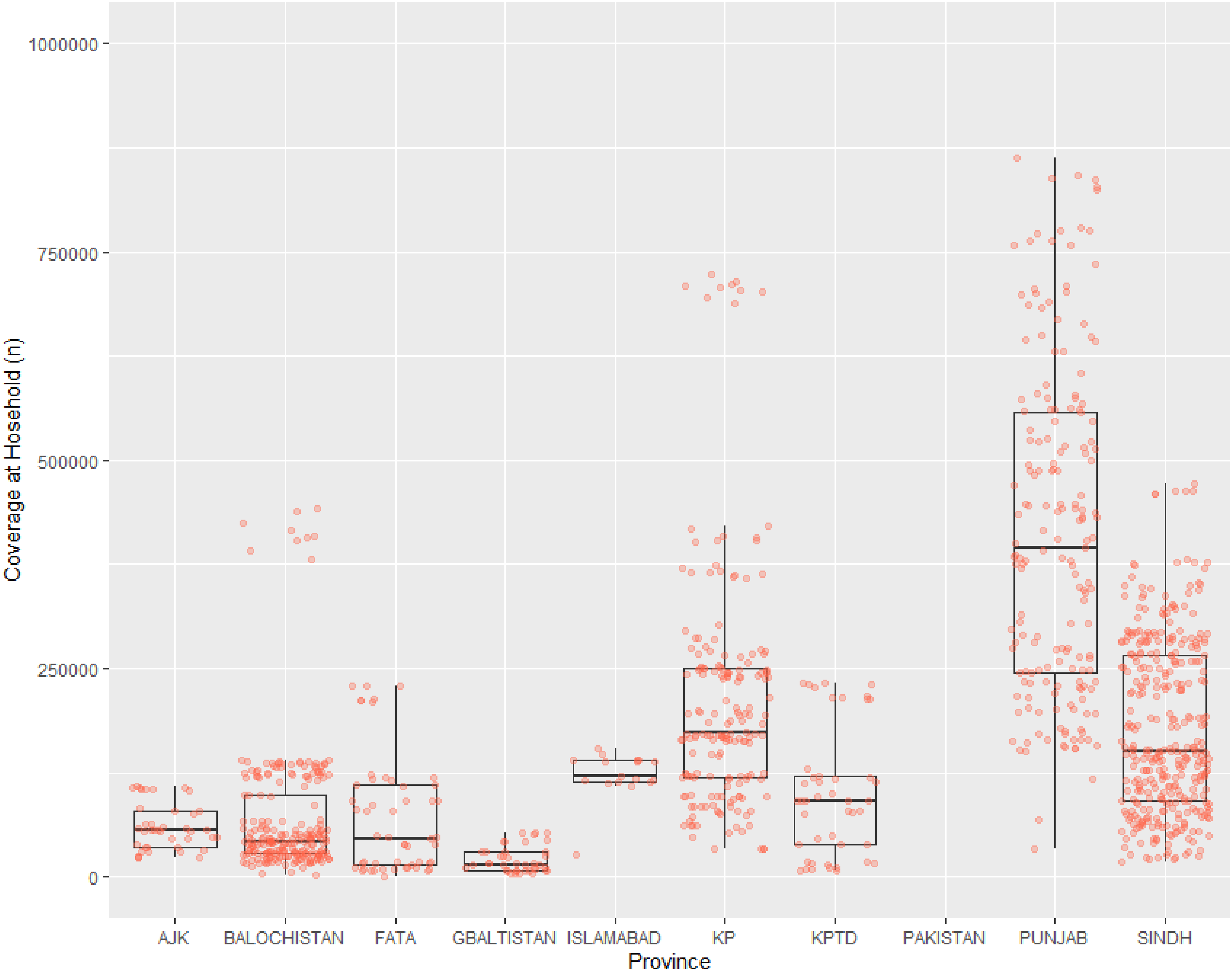


1000000
750000
500000
250000
0
Coverage at Hosehold (n)
AJK
BALOCHISTAN
FATA
GBALTISTAN
ISLAMABAD
KP
KPTD
PAKISTAN
PUNJAB
SINDH
Province

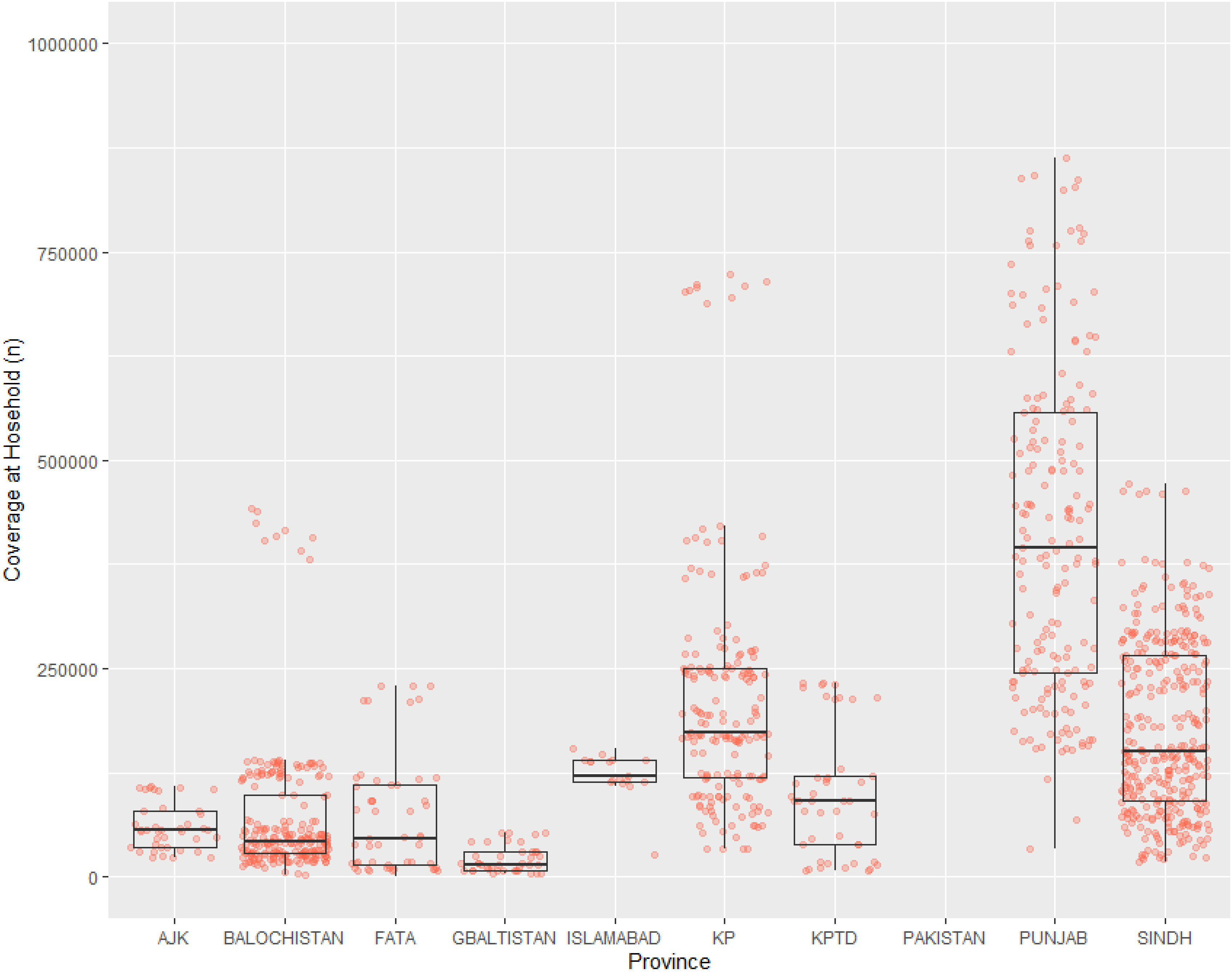
- By adding points to boxplot, we can have a better idea of the number of measurements and of their distribution:



```
ggplot(data = dat,
       mapping = aes(x = Province, y = `Coverage at Hosehold (n)`)) +
  geom_boxplot(alpha = 0) +
  geom_jitter(alpha = 0.3, colour = "tomato") +
  ylim(0, 1e+6)
```


1000000
750000
500000
250000
0
Coverage at Hosehold (n)
AJK
BALOCHISTAN
FATA
GBALTISTAN
ISLAMABAD
KP
KPTD
PAKISTAN
PUNJAB
SINDH
Province

- Notice how the boxplot layer is behind the jitter layer?
- What do you need to change in the code to put the boxplot in front of the points such that it’s not hidden?

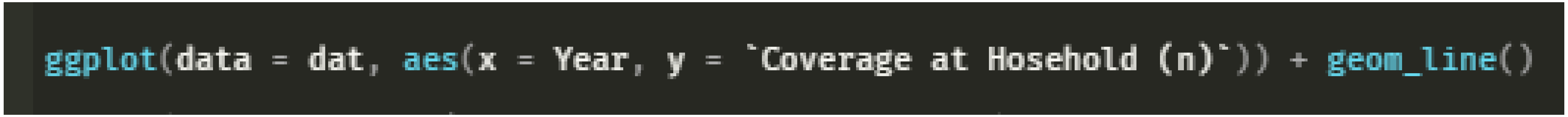

Coverage at Hosehold (n)
1000000
750000
500000
250000
0
AJK
BALOCHISTAN
FATA
GBALTISTAN
ISLAMABAD
KP
KPTD
PAKISTAN
PUNJAB
SINDH
Province

Your turn

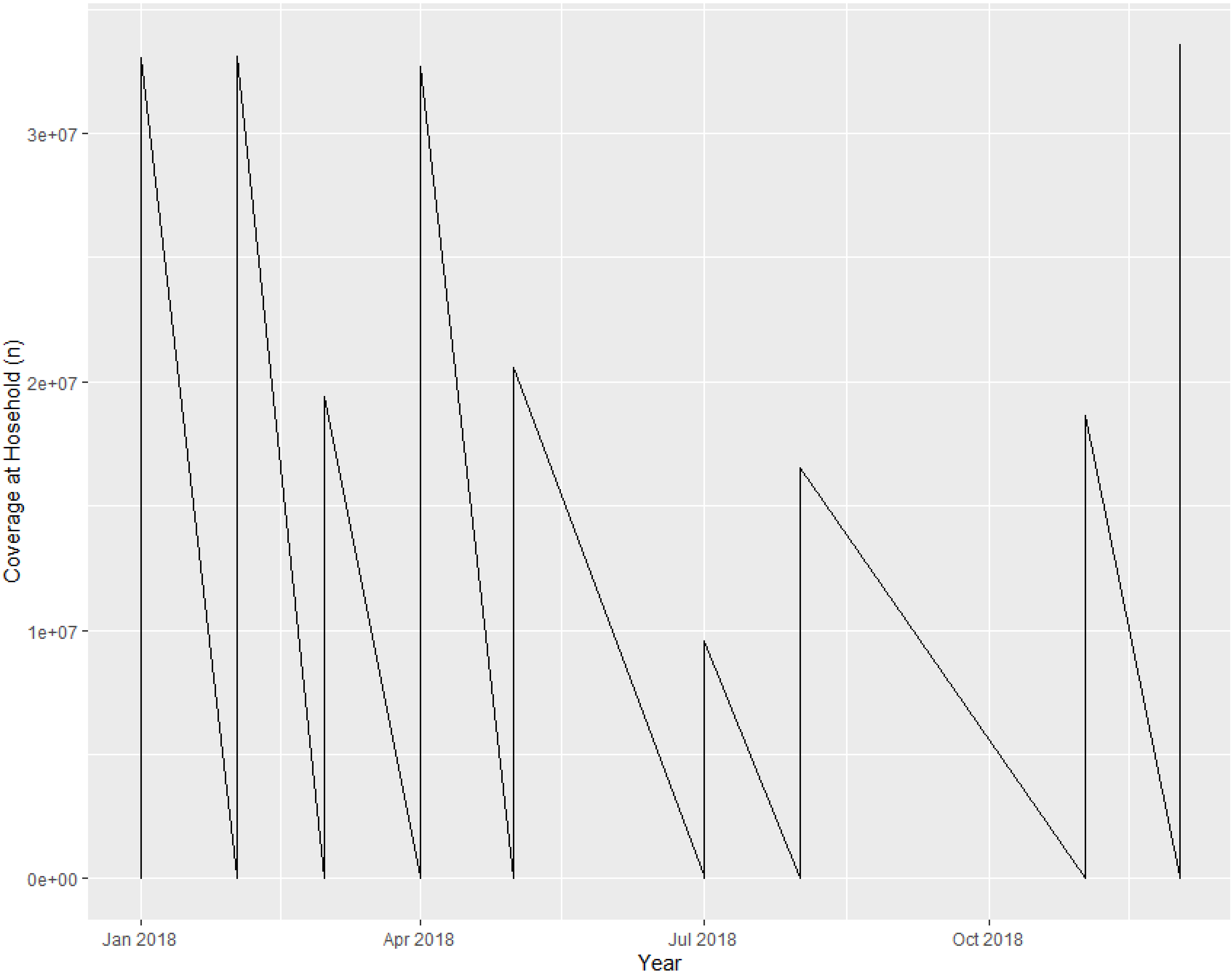
- Boxplots are useful summaries, but hide the shape of the distribution. For example, if the distribution is bimodal, we would not see it in a boxplot. An alternative to the boxplot is the violin plot, where the shape (of the density of points) is drawn.
- Replace the box plot with a violin plot; see geom_violin().
- In many types of data, it is important to consider the scale of the observations. For example, it may be worth changing the scale of the axis to better distribute the observations in the space of the plot. Changing the scale of the axes is done similarly to adding/modifying other components (i.e., by incrementally adding commands). Try making these modifications:
- Represent weight on the log10 scale; see scale_y_log10().
- So far, we've looked at the distribution of xxx within species. Try making a new plot to explore the distribution of another variable within each species.
- Create a boxplot for xxx. Overlay the boxplot layer on a jitter layer to show actual measurements.
- Add colour to the data points on your boxplot according to the plot from which the sample was taken (Year).
- Hint: Check the class for Year. Consider changing the class of Year from integer to factor. Why does this change how R makes the graph?

Plotting time series data

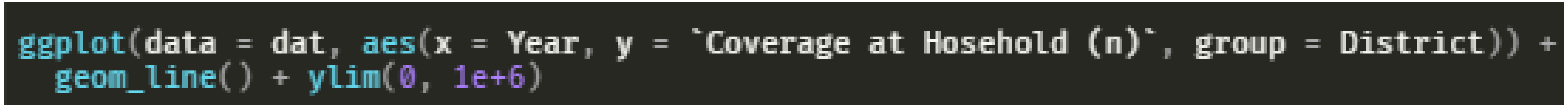
- Let’s visualise the number of coverage per household per year for each District as line plot



```
ggplot(data = dat, aes(x = Year, y = `Coverage at Hosehold (n)`)) + geom_line()
```

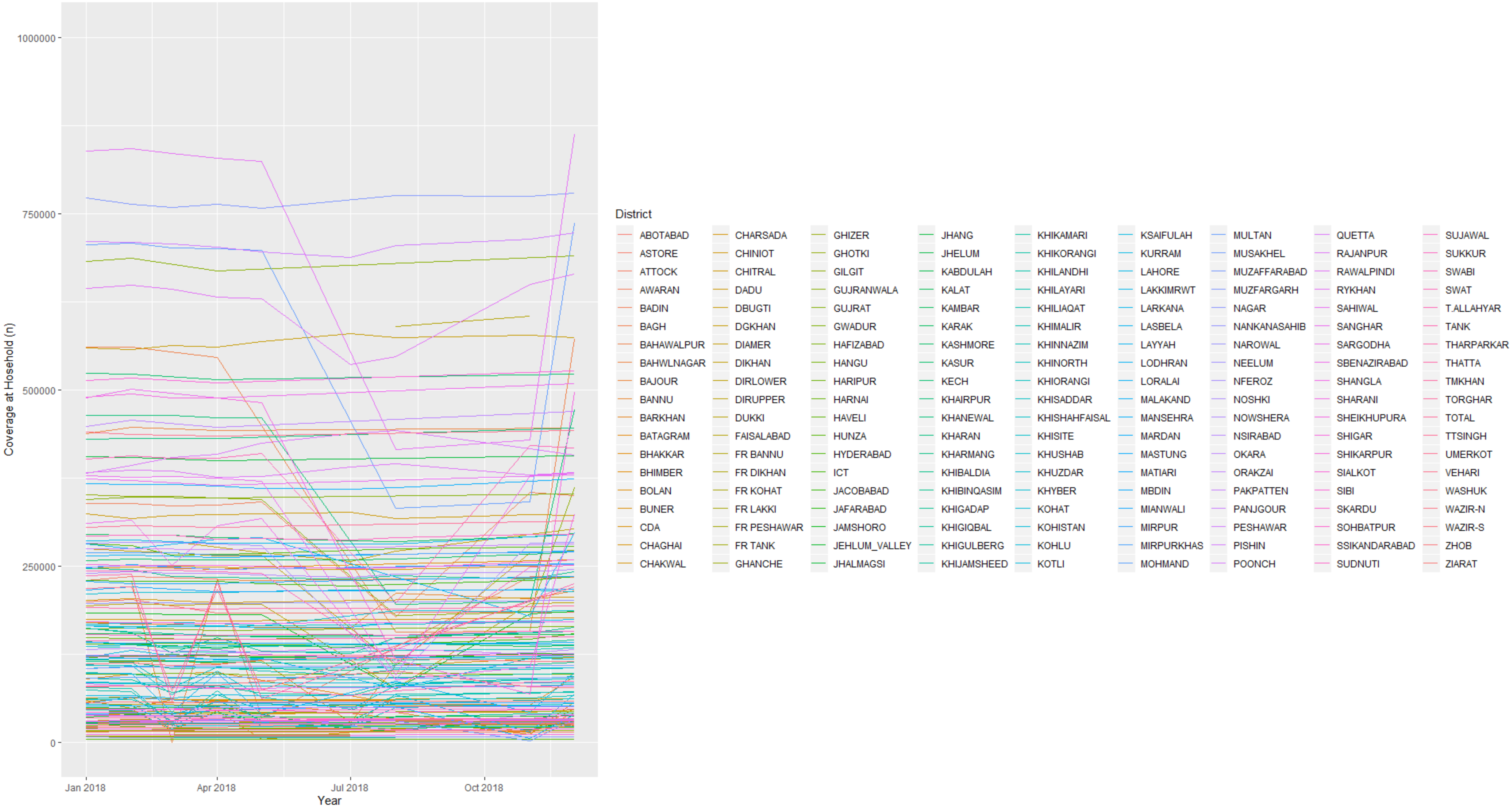

Coverage at Hosehold (n)
3e+07
2e+07
1e+07
0e+00
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Year

- Unfortunately, this does not work because we plotted data for all the Districts together.
- We need to tell ggplot to draw a line for each by modifying the aesthetic function to include group = District:



```
ggplot(data = dat, aes(x = Year, y = `Coverage at Hosehold (n)`, group = District)) +
  geom_line() + ylim(0, 1e+6)
```

- We will be able to distinguish District in the plot if we add colours (using colour also automatically groups the data):


Coverage at Hosehold (n)
1000000
750000
500000
250000
0
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Year
District
ABOTABAD
ASTORE
ATTOCK
AWARAN
BADIN
BAGH
BAHAWALPUR
BAHWLNAGAR
BAJOUR
BANNU
BARKHAN
BATAGRAM
BHAKKAR
BHIMBER
BOLAN
BUNER
CDA
CHAGHAI
CHAKWAL
CHARSADA
CHINIOT
CHITRAL
DADU
DBUGTI
DGKHAN
DIAMER
DIKHAN
DIRLOWER
DIRUPPER
DUKKI
FAISALABAD
FR BANNU
FR DIKHAN
FR KOHAT
FR LAKKI
FR PESHAWAR
FR TANK
GHANCHE
GHIZER
GHOTKI
GILGIT
GUJRANWALA
GUJRAT
GWADUR
HAFIZABAD
HANGU
HARIPUR
HARNAI
HAVELI
HUNZA
HYDERABAD
ICT
JACOBABAD
JAFARABAD
JAMSHORO
JEHLUM_VALLEY
JHALMAGSI
JHANG
JHELUM
KABDULLAH
KALAT
KAMBAR
KARAK
KASHMORE
KASUR
KECH
KHAIRPUR
KHANEWAL
KHARAN
KHARMANG
KHIBALDIA
KHIBINQASIM
KHIGADAP
KHIGIQBAL
KHIGULBERG
KHIJAMSHEED
KHIKAMARI
KHIKORANGI
KHILANDHI
KHILAYARI
KHILIAQAT
KHIMALIR
KHINNAZIM
KHINORTH
KHIORANGI
KHISADDAR
KHISHAHFAISAL
KHISITE
KHUSHAB
KHUZDAR
KHYBER
KOHAT
KOHISTAN
KOHLU
KOTLI
KSAIFULAH
KURRAM
LAHORE
LAKKIMRWT
LARKANA
LASBELA
LAYYAH
LODHRAN
LORALAI
MALAKAND
MANSEHRA
MARDAN
MASTUNG
MATIARI
MBDIN
MIANWALI
MIRPUR
MIRPURKHAS
MOHMAND
MULTAN
MUSAKHEL
MUZAFFARABAD
MUZFARGARH
NAGAR
NANKANASAHIB
NAROWAL
NEELUM
NFEROZ
NOSHKI
NOWSHERA
NSIRABAD
OKARA
ORAKZAI
PAKPATTEN
PANJGOUR
PESHAWAR
PISHIN
POONCH
QUETTA
RAJANPUR
RAWALPINDI
RYKHAN
SAHIWAL
SANGHAR
SARGODHA
SBENAZIRABAD
SHANGLA
SHARANI
SHEIKHUPURA
SHIGAR
SHIKARPUR
SIALKOT
SIBI
SKARDU
SOHBATPUR
SSIKANDARABAD
SUDNUTI
SUJAWAL
SUKKUR
SWABI
SWAT
T.ALLAHYAR
TANK
THARPARKAR
THATTA
TMKHAN
TORGHAR
TOTAL
TTSINGH
UMERKOT
VEHARI
WASHUK
WAZIR-N
WAZIR-S
ZHOB
ZIARAT

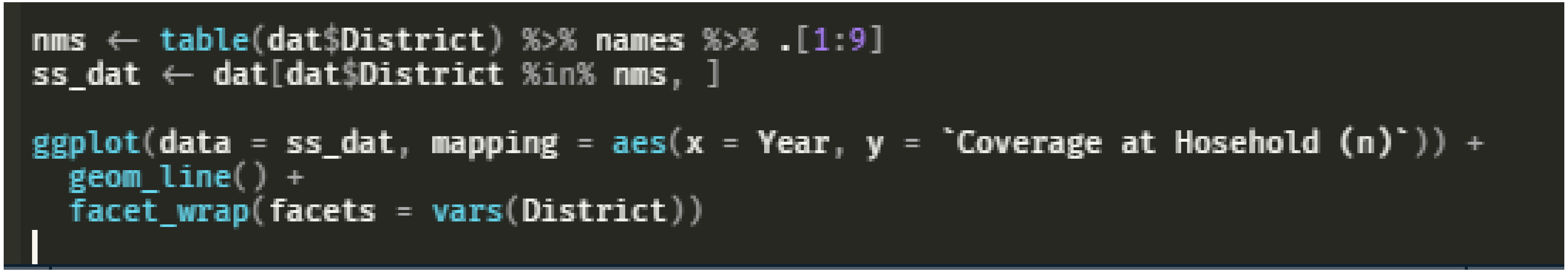
- Both geometries allow to to specify faceting variables specified within vars().
- Eg

facet_wrap(facets = vars(facet_variable)) or facet_grid(rows = vars(row_variable), cols = vars(col_variable)).

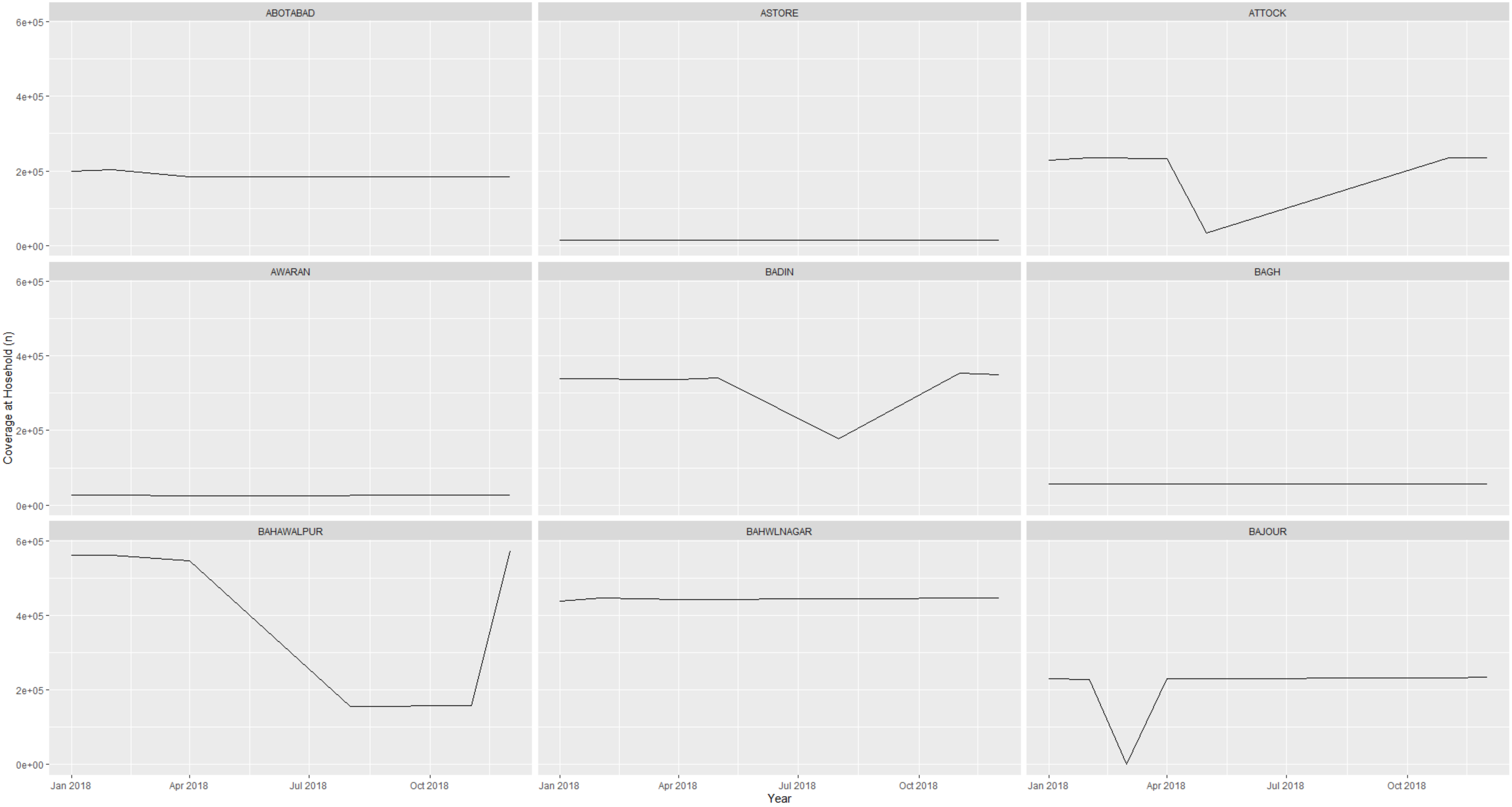
- Let’s start by using facet_wrap() to make a time series plot for a subset of Districts:

Faceting

- ggplot2 has a special technique called faceting that allows the user to split one plot into multiple plots based on a factor included in the dataset.
- There are two types of facet functions:
    - facet_wrap() arranges a one-dimensional sequence of panels to allow them to cleanly fit on one page.
    - facet_grid() allows you to form a matrix of rows and columns of panels.



```
nms ← table(dat$District) %>% names %>% .[1:9]
ss_dat ← dat[dat$District %in% nms, ]

ggplot(data = ss_dat, mapping = aes(x = Year, y = `Coverage at Hosehold (n)`)) +
  geom_line() +
  facet_wrap(facets = vars(District))
```

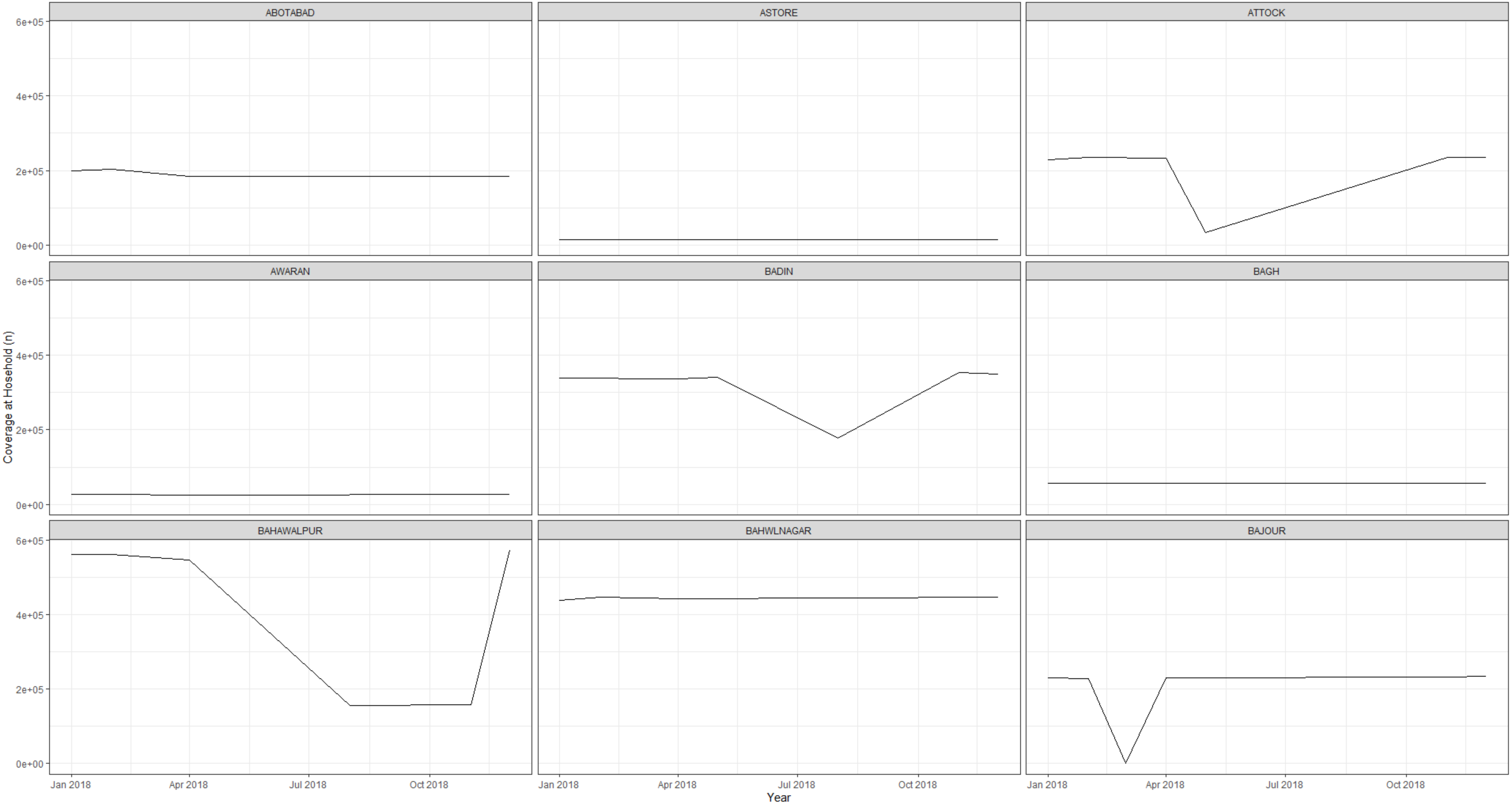

ABOTABAD
ASTORE
ATTOCK
AWARAN
BADIN
BAGH
BAHAWALPUR
BAHWLNAGAR
BAJOUR
Coverage at Hosehold (n)
6e+05
4e+05
2e+05
0e+00
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Year

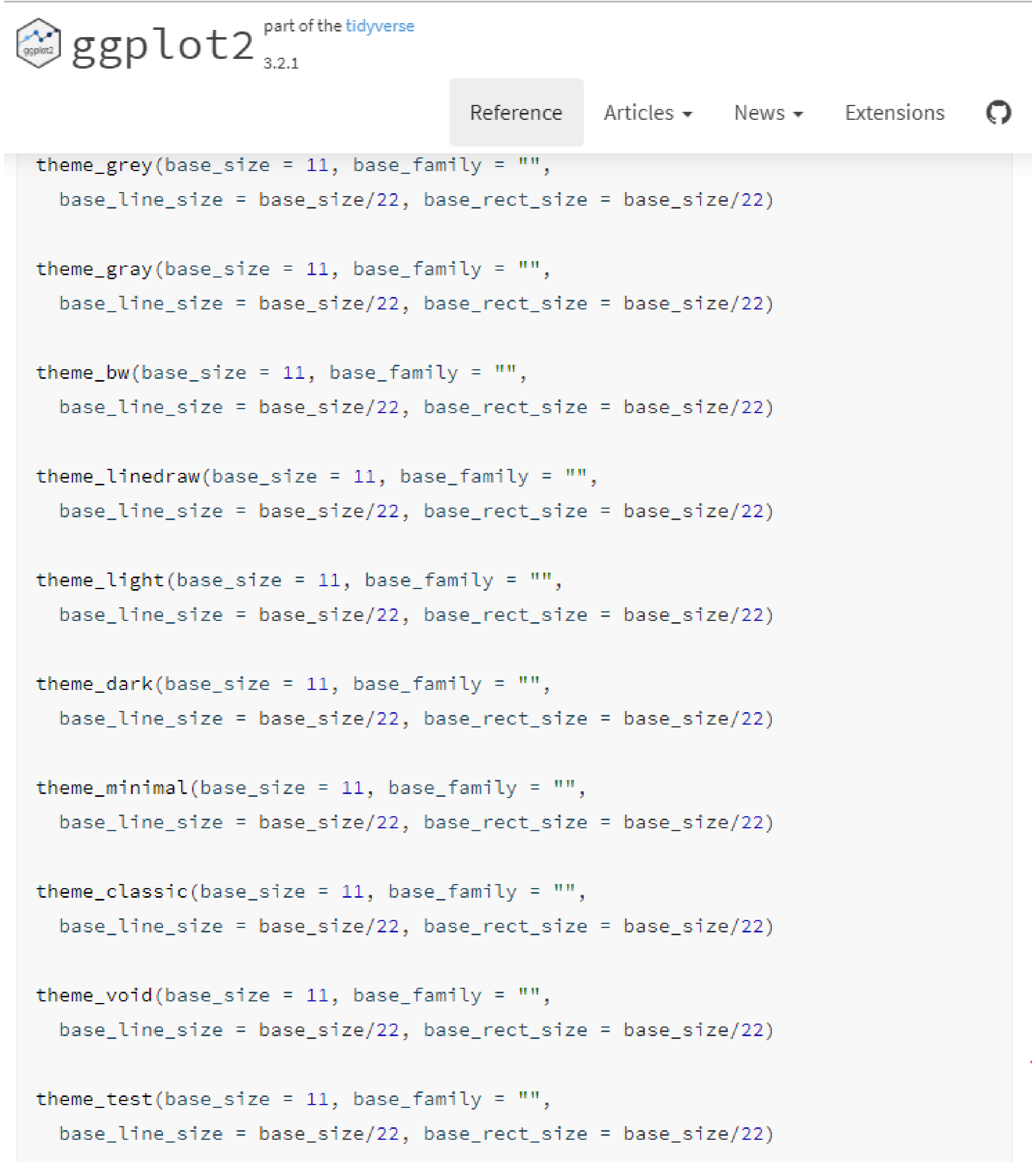
Note: In earlier versions of `ggplot2` you need to use an interface using formulas to specify how plots are faceted (and this is still supported in new versions). The equivalent syntax is:

```
# facet wrap
facet_wrap(vars(genus))    # new
facet_wrap(~ genus)        # old

# grid on both rows and columns
facet_grid(rows = vars(genus), cols = vars(sex))   # new
facet_grid(genus ~ sex)                            # old

# grid on rows only
facet_grid(rows = vars(genus))   # new
facet_grid(genus ~ .)            # old

# grid on columns only
facet_grid(cols = vars(genus))   # new
facet_grid(. ~ genus)            # old
```

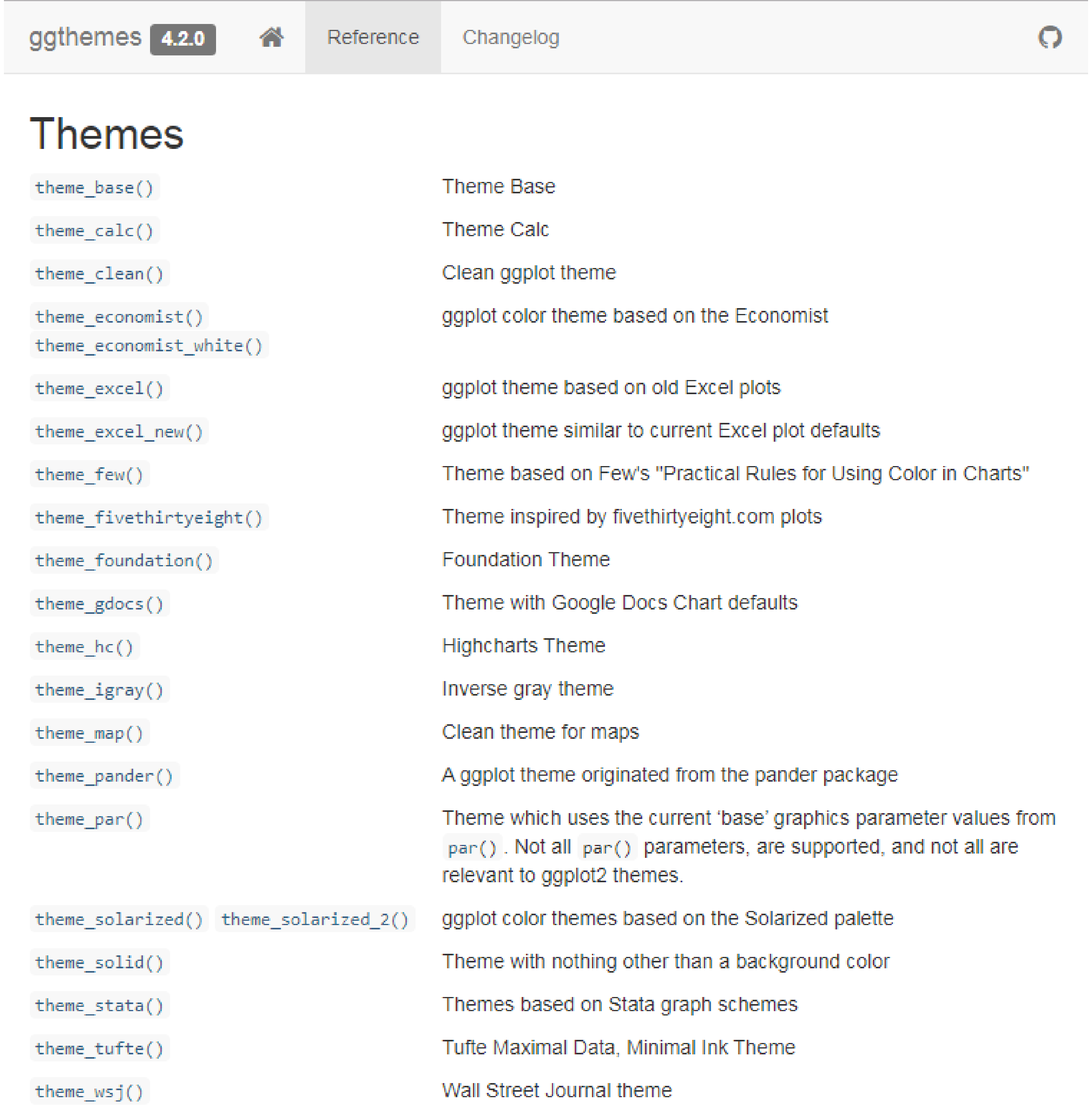
ggplot themes

- Usually plots with white background look more readable when printed.
- Every single component of a ggplot graph can be customized using the generic theme() function, as we will see below.
- However, there are pre-loaded themes available that change the overall appearance of the graph without much effort.
- Eg
  - we can change our previous graph to have a simpler white background using the theme_bw() function:


ABOTABAD
ASTORE
ATTOCK
AWARAN
BADIN
BAGH
BAHAWALPUR
BAHWLNAGAR
BAJOUR
6e+05
4e+05
2e+05
0e+00
Coverage at Hosehold (n)
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Year



```
theme_grey(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_gray(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_bw(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_linedraw(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_light(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_dark(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_minimal(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_classic(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_void(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)

theme_test(base_size = 11, base_family = "",
  base_line_size = base_size/22, base_rect_size = base_size/22)
```



Themes

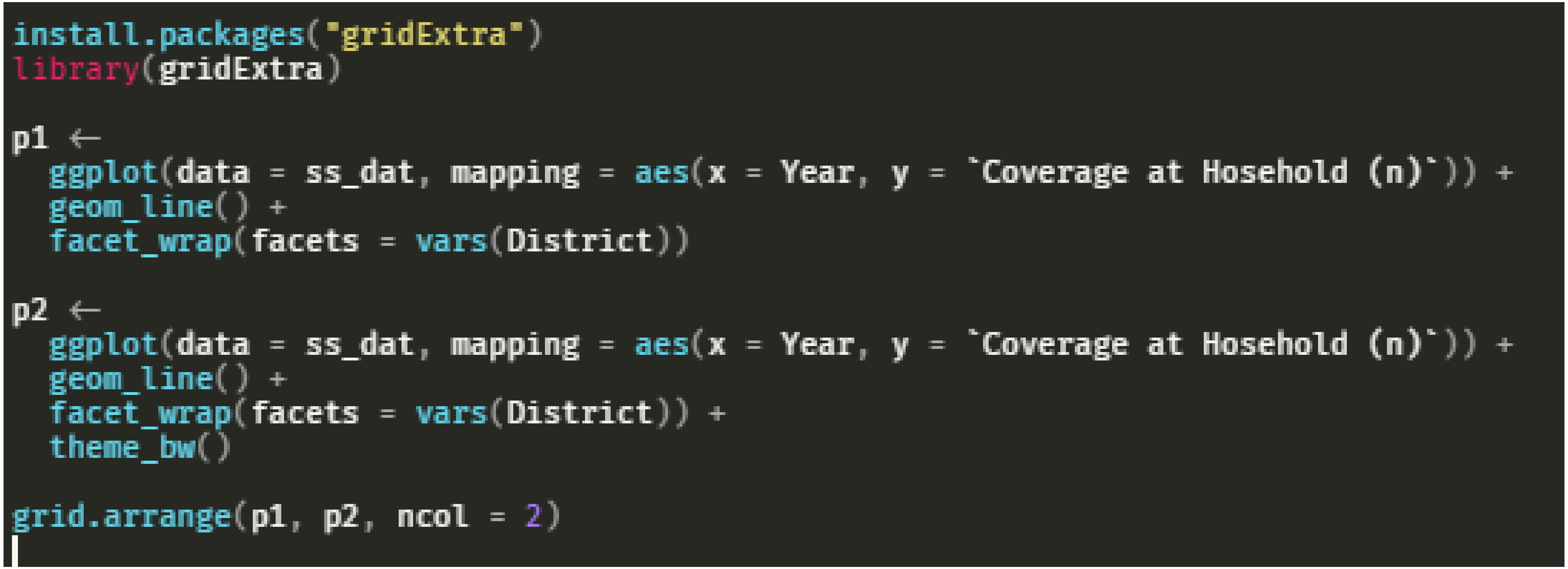
`theme_base()`	Theme Base
`theme_calc()`	Theme Calc
`theme_clean()`	Clean ggplot theme

`theme_economist()`
`theme_economist_white()`
	ggplot color theme based on the Economist

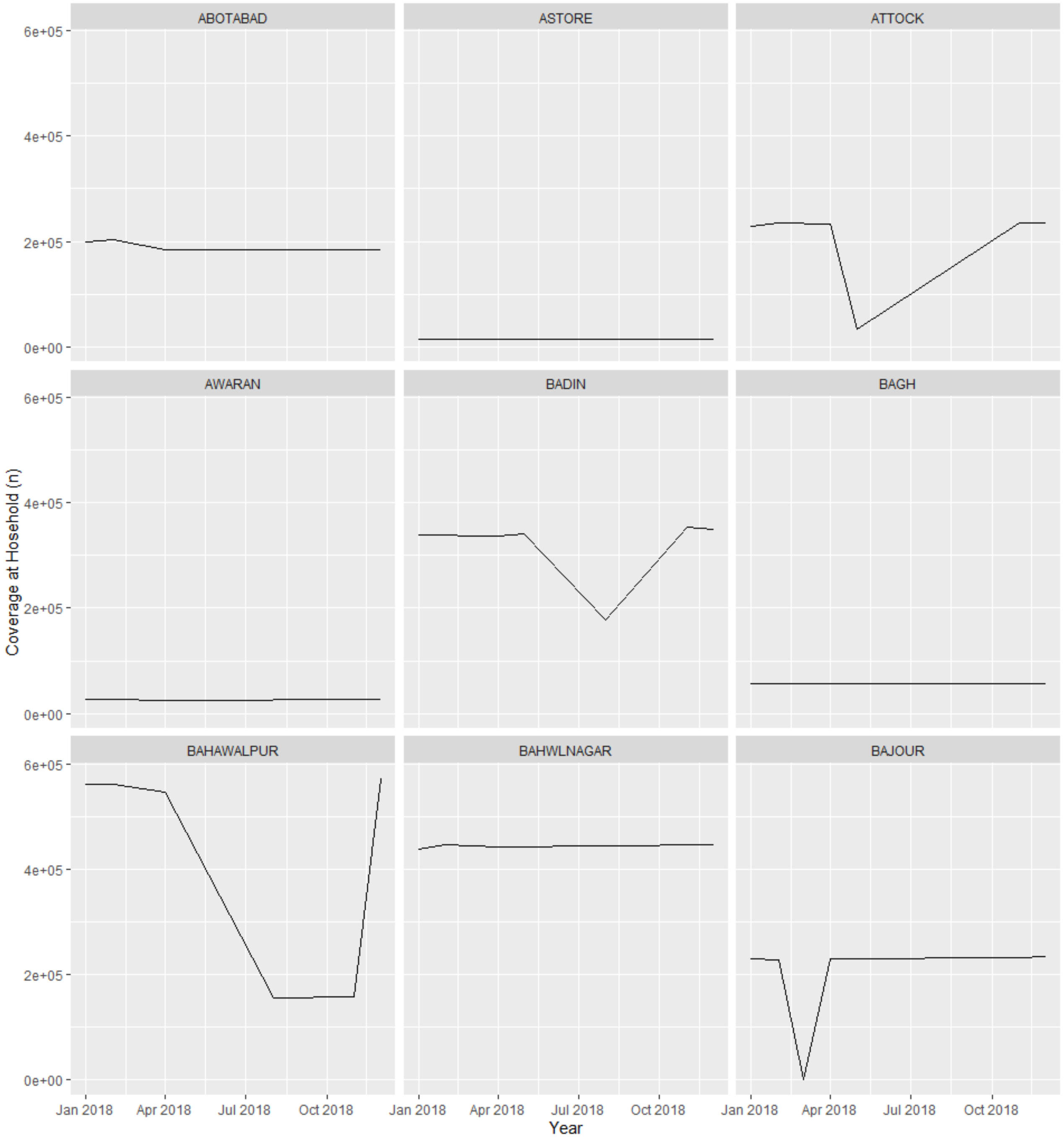
`theme_excel()`	ggplot theme based on old Excel plots
`theme_excel_new()`	ggplot theme similar to current Excel plot defaults
`theme_few()`	Theme based on Few's "Practical Rules for Using Color in Charts"
`theme_fivethirtyeight()`	Theme inspired by fivethirtyeight.com plots
`theme_foundation()`	Foundation Theme
`theme_gdocs()`	Theme with Google Docs Chart defaults
`theme_hc()`	Highcharts Theme
`theme_igray()`	Inverse gray theme
`theme_map()`	Clean theme for maps
`theme_pander()`	A ggplot theme originated from the pander package
`theme_par()`	Theme which uses the current 'base' graphics parameter values from `par()` . Not all `par()` parameters, are supported, and not all are relevant to ggplot2 themes.
`theme_solarized()` `theme_solarized_2()`	ggplot color themes based on the Solarized palette
`theme_solid()`	Theme with nothing other than a background color
`theme_stata()`	Themes based on Stata graph schemes
`theme_tufte()`	Tufte Maximal Data, Minimal Ink Theme
`theme_wsj()`	Wall Street Journal theme

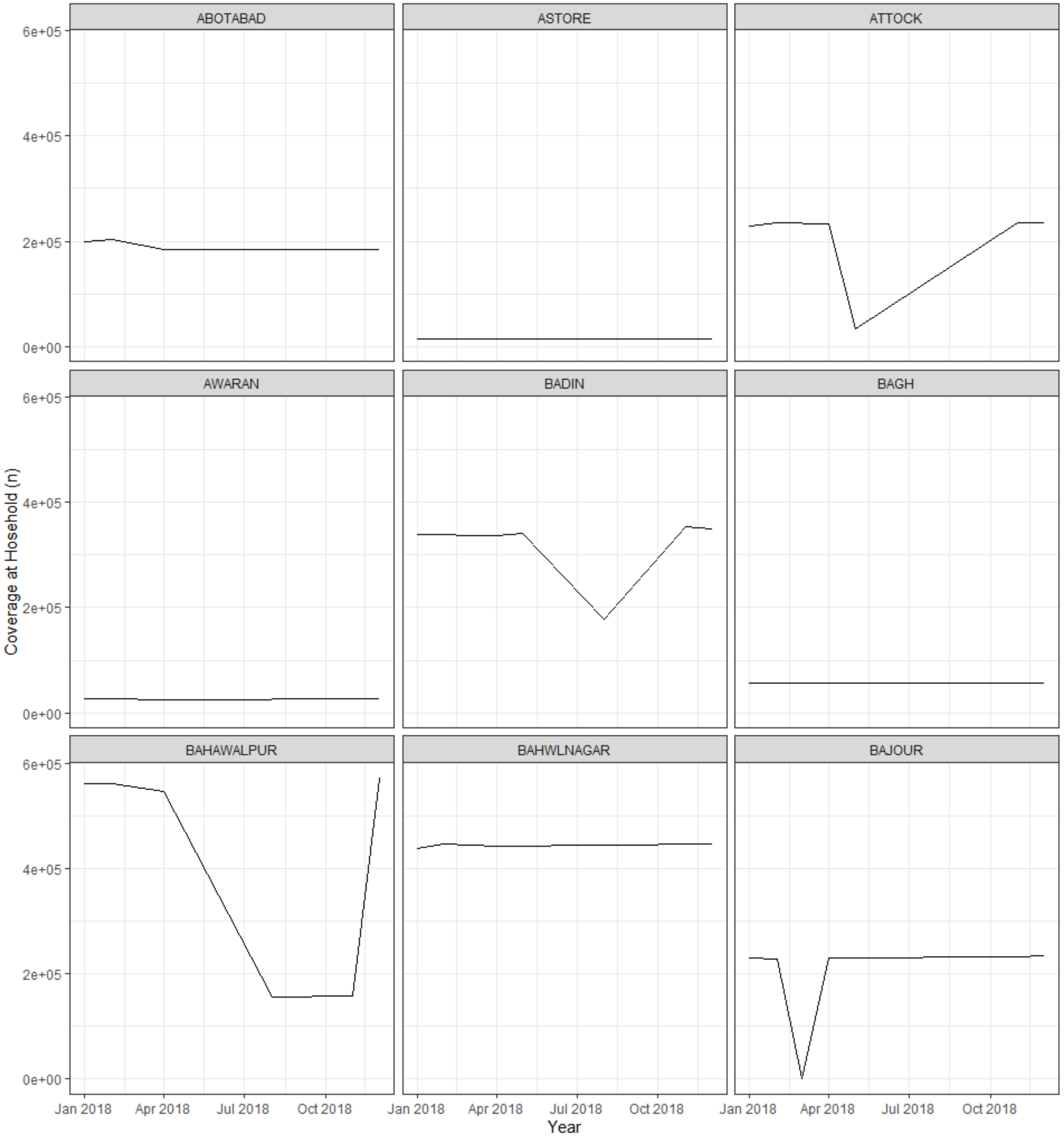
Arranging and exporting plots

- Faceting is a great tool for splitting one plot into multiple plots, but sometimes you may want to produce a single figure that contains multiple plots using different variables or even different data frames. The gridExtra package allows us to combine separate ggplots into a single figure using grid.arrange():



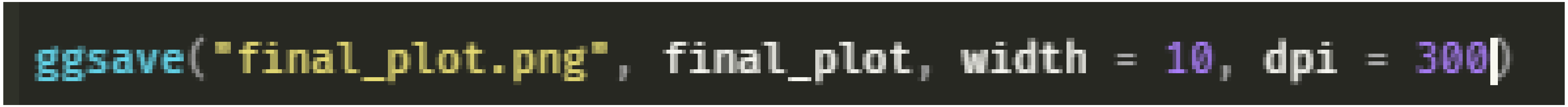
```
install.packages("gridExtra")
library(gridExtra)

p1 <-
  ggplot(data = ss_dat, mapping = aes(x = Year, y = `Coverage at Hosehold (n)`)) +
  geom_line() +
  facet_wrap(facets = vars(District))

p2 <-
  ggplot(data = ss_dat, mapping = aes(x = Year, y = `Coverage at Hosehold (n)`)) +
  geom_line() +
  facet_wrap(facets = vars(District)) +
  theme_bw()

grid.arrange(p1, p2, ncol = 2)
```


ABOTABAD
ASTORE
ATTOCK
AWARAN
BADIN
BAGH
BAHAWALPUR
BAHWLNAGAR
BAJOUR
6e+05
4e+05
2e+05
0e+00
Coverage at Hosehold (n)
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Year


ABOTABAD
ASTORE
ATTOCK
AWARAN
BADIN
BAGH
BAHAWALPUR
BAHWLNAGAR
BAJOUR
6e+05
4e+05
2e+05
0e+00
Coverage at Hosehold (n)
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Year

Saving plots

- use the ggsave() function, which allows you easily change the dimension and resolution of your plot by adjusting the appropriate arguments (width, height and dpi).



```
ggsave("final_plot.png", final_plot, width = 10, dpi = 300)
```