Learning *R*

Carl James Schwarz

StatMathComp Consulting by Schwarz
cschwarz.stat.sfu.ca @ gmail.com

Plotting with ggplot2 - Advanced

Table of Contents I

Advanced Plotting using *ggplot2*

Facetting

Plotting with *ggplot2* - Facetting

Make separate panels by one or two variables.

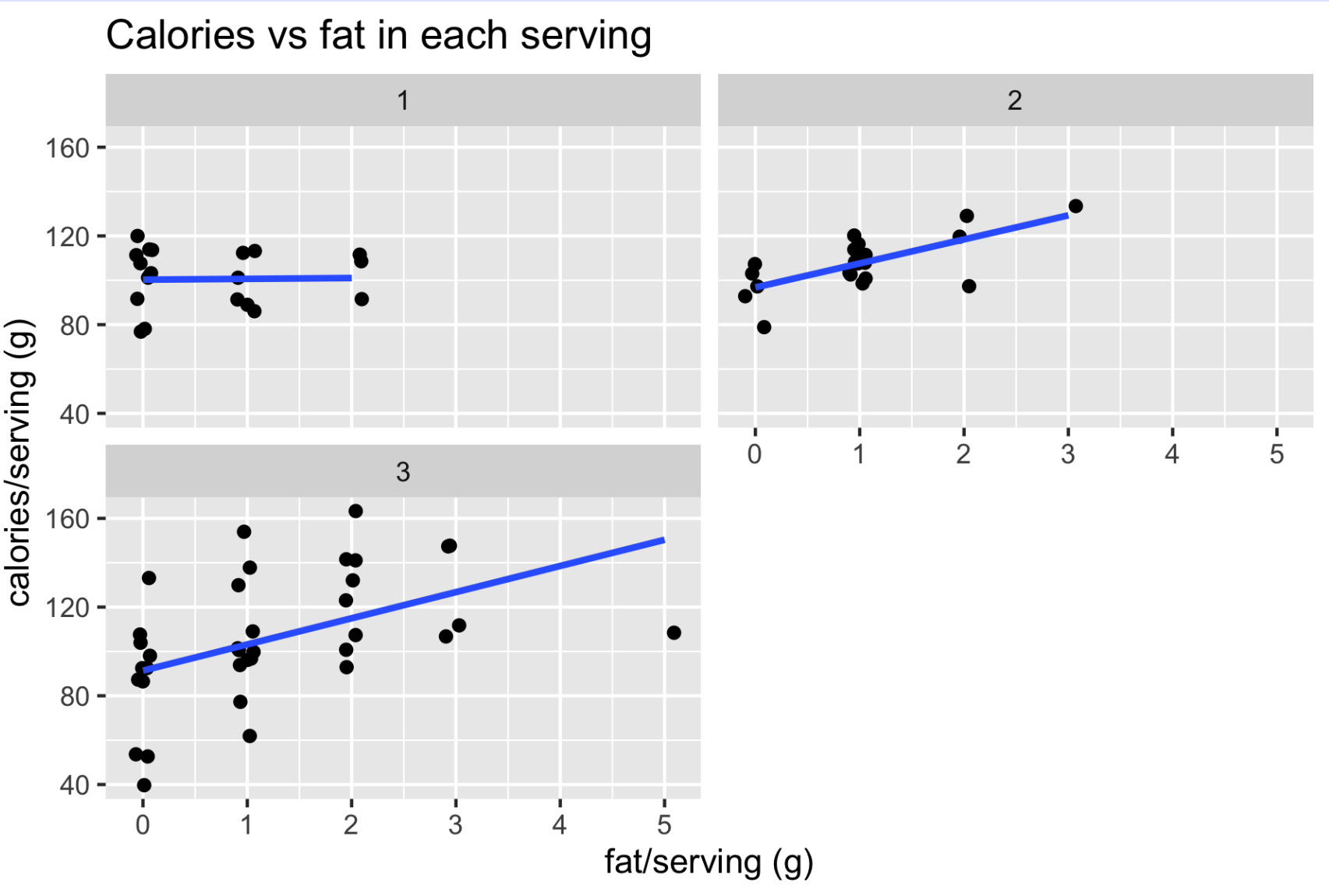
... facet_wrap($\sim$ v2 , ncol=2)

... facet_grid(v1 $\sim$ v2)

Scales can be free or fixed on both X and Y axes.

```
cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))+
     ggtitle("Calories vs fat in each serving")+
     xlab("Fat/serving (g)")+ylab("Calories/serving (g)")+
     geom_jitter()+
     geom_smooth(method="lm", se=FALSE)+
     facet_wrap(~shelfF, ncol=2)
 cerealplot
```

Plotting with *ggplot2* - Facetting



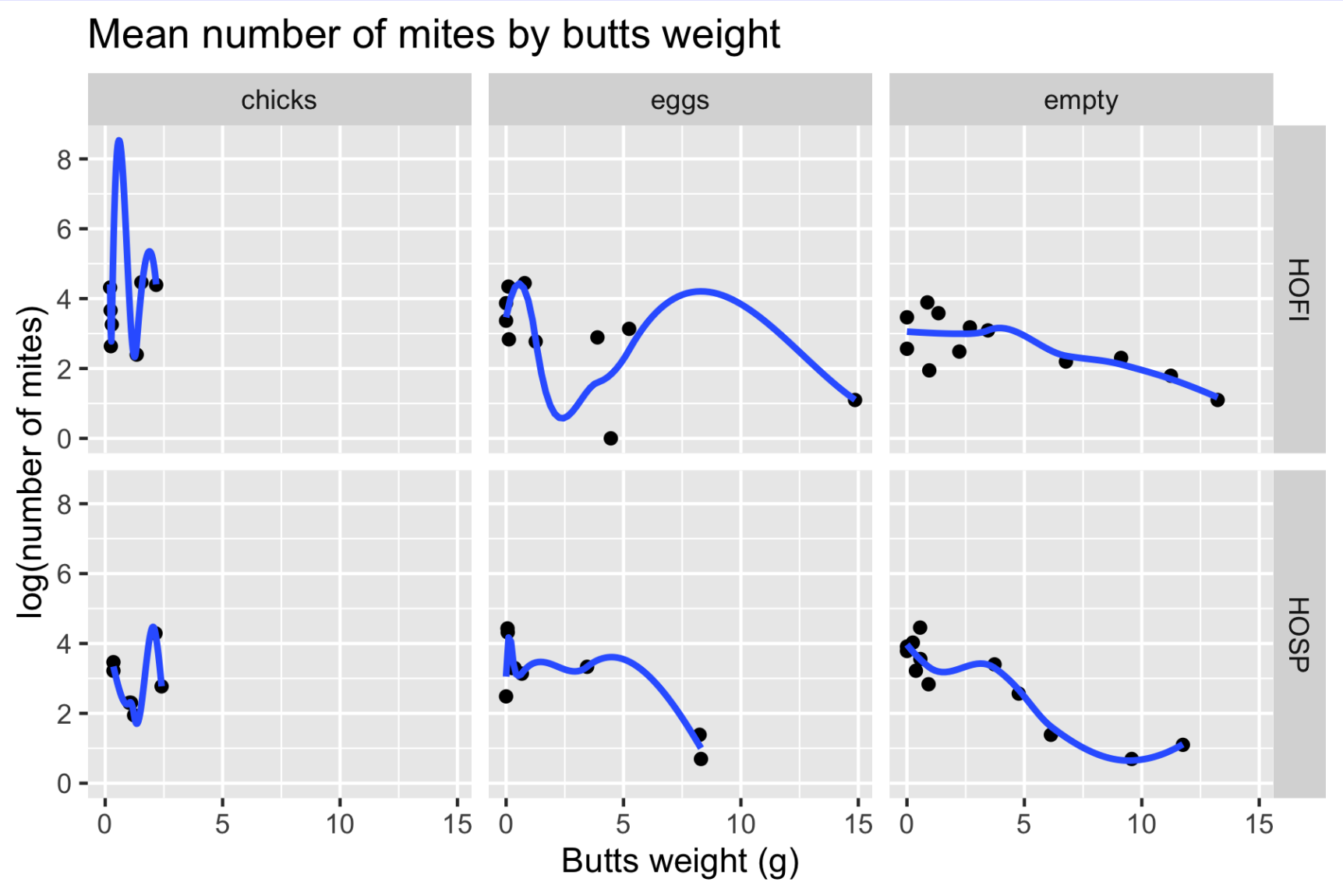
Plotting with *ggplot2* - Facetting - Exercise

Plot the log(*number mites*) vs. butts weight with fitted line for each combination of species and nest content.

Plotting with *ggplot2* - Facetting - Exercise

```
1 mitesfacet <- ggplot(data=butts, aes(x=Butts.weight, y=log.m
2     ggtitle("Mean number of mites by butts weight")+
3     xlab("Butts weight (g)")+ylab("log(number of mites)")+
4     geom_point()+
5     geom_smooth(method="loess", se=FALSE)+
6     facet_grid(Species ~ Nest.content)
7 mitesfacet
```

Plotting with *ggplot2* - Facetting - Exercise

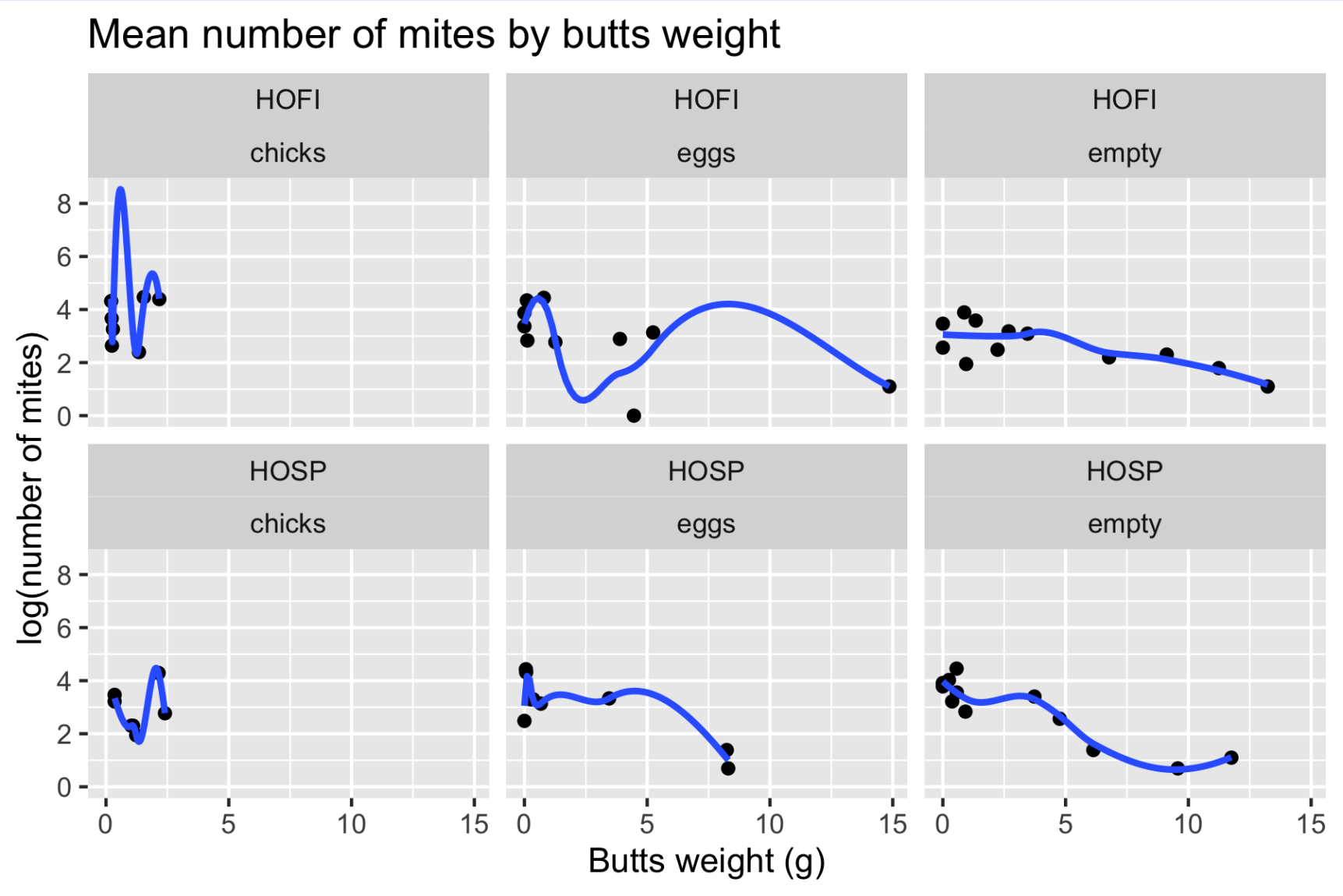


Plotting with *ggplot2* - Facetting

More than one variable can used in a facet dimension.

```
1 mitesfacet <- ggplot(data=butts, aes(x=Butts.weight, y=log.m
2     ggtitle("Mean number of mites by butts weight")+
3     xlab("Butts weight (g)")+ylab("log(number of mites)")+
4     geom_point()+
5     geom_smooth(method="loess", se=FALSE)+
6     facet_wrap( ~Species + Nest.content, ncol=3)
7 mitesfacet
```

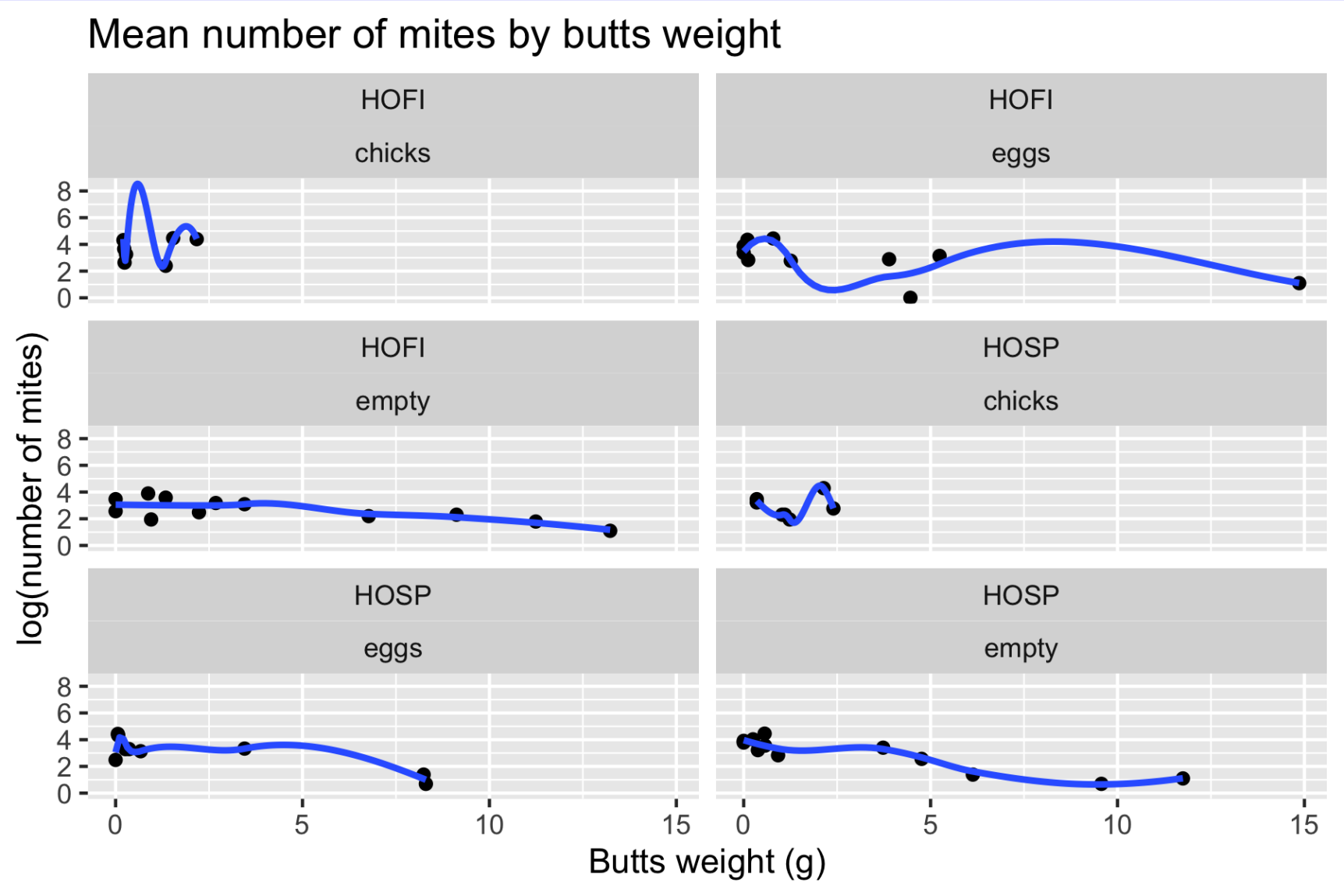
Plotting with *ggplot2* - Facetting - Exercise



Plotting with *ggplot2* - Facetting

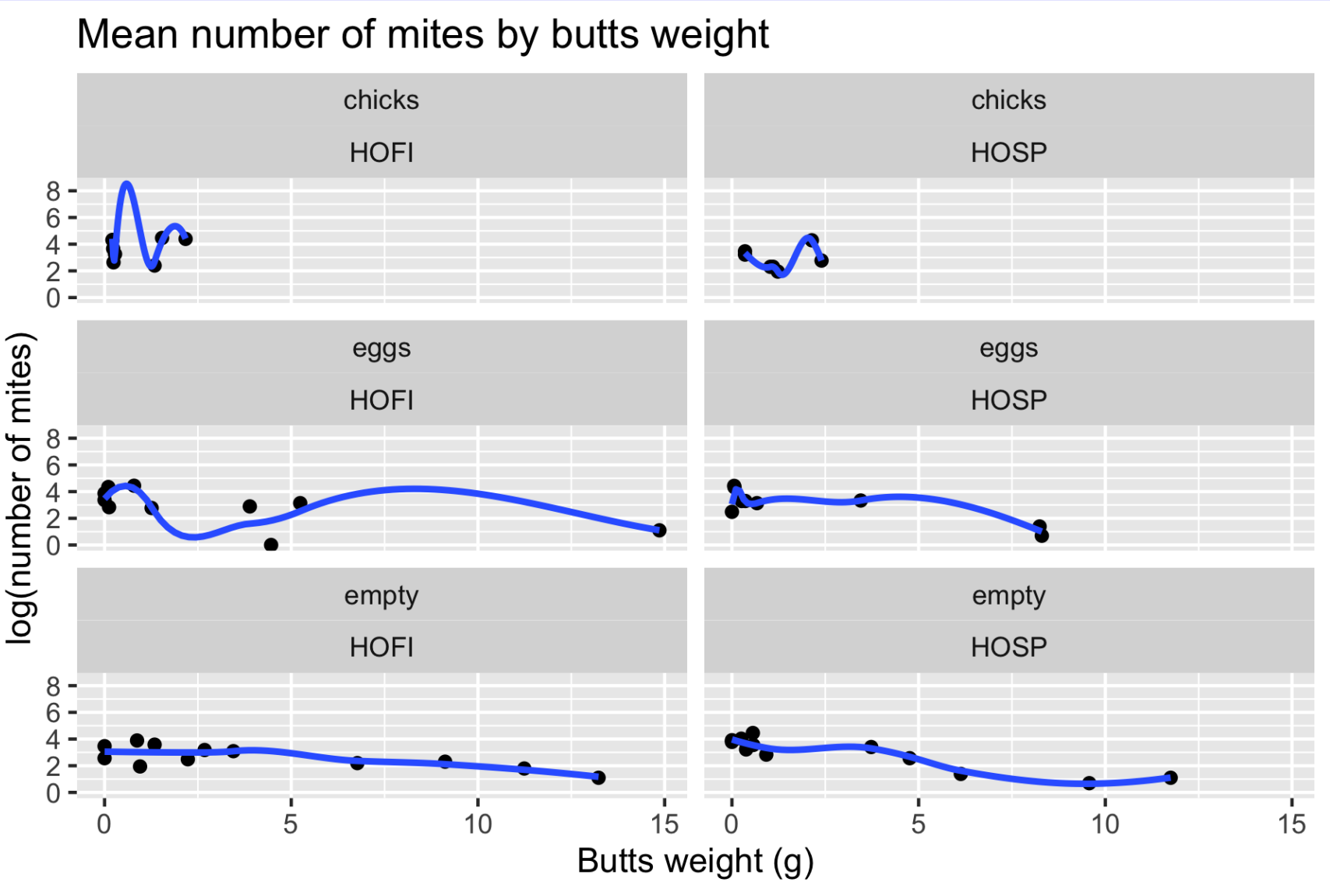
More than one variable can used in a facet dimension.
Repeat with different values for *ncol* and different order of facet variables.

Plotting with *ggplot2* - Facetting



Plotting with *ggplot2* - Facetting

Notice impact of changing order of variables in *facet_wrap()* function.



Facet labels can be modified using the *labeller* argument:

... facet_wrap(v1 ~ . , nrow=2, labeller=....)
where

- label_value - display value of factor
- label_both - display variable name and value
- label_parsed - useful for math expressions

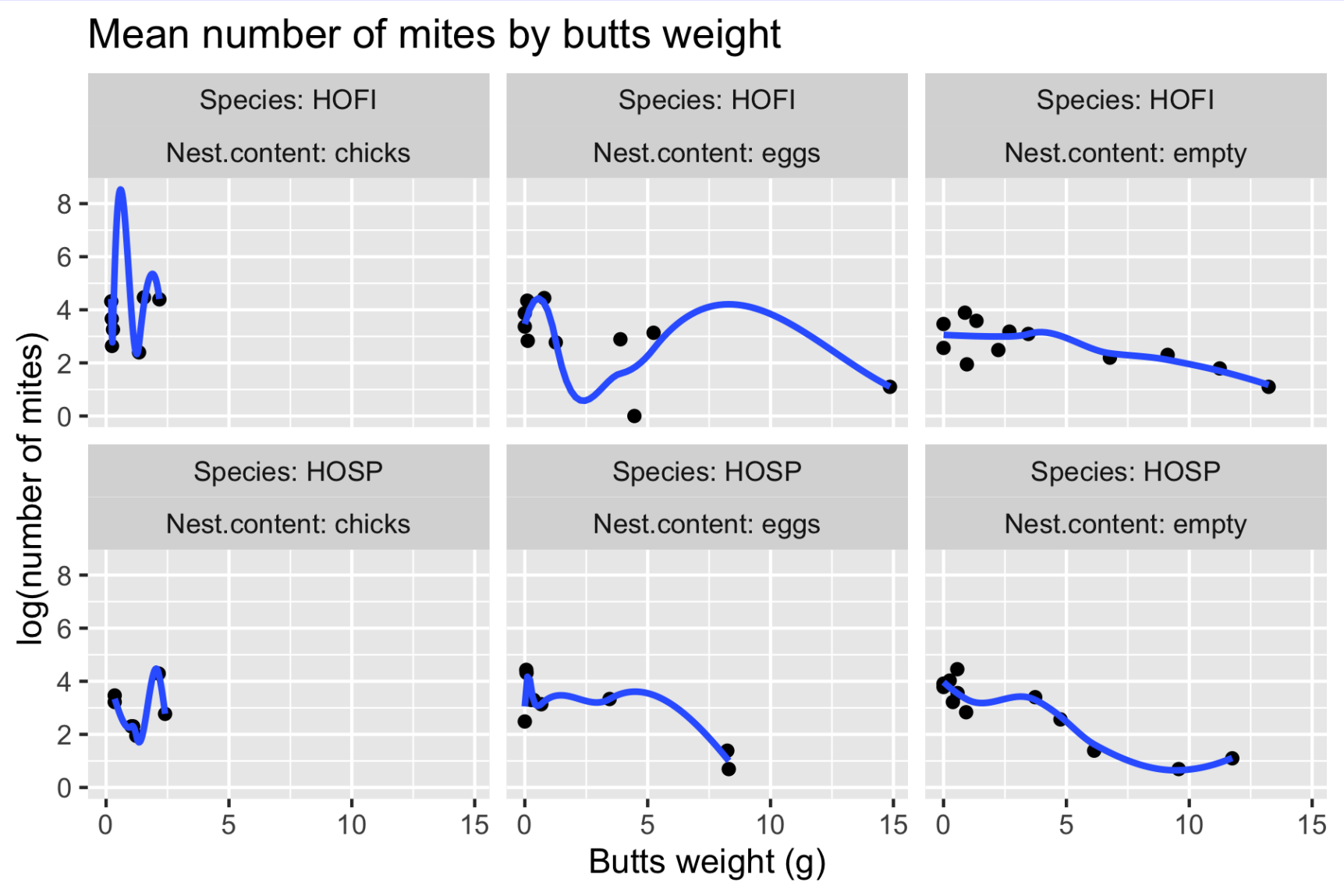
....

See
`http://ggplot2.tidyverse.org/reference/labellers.html`
for more details

Plotting with *ggplot2* - Facetting - Modifying facet labels

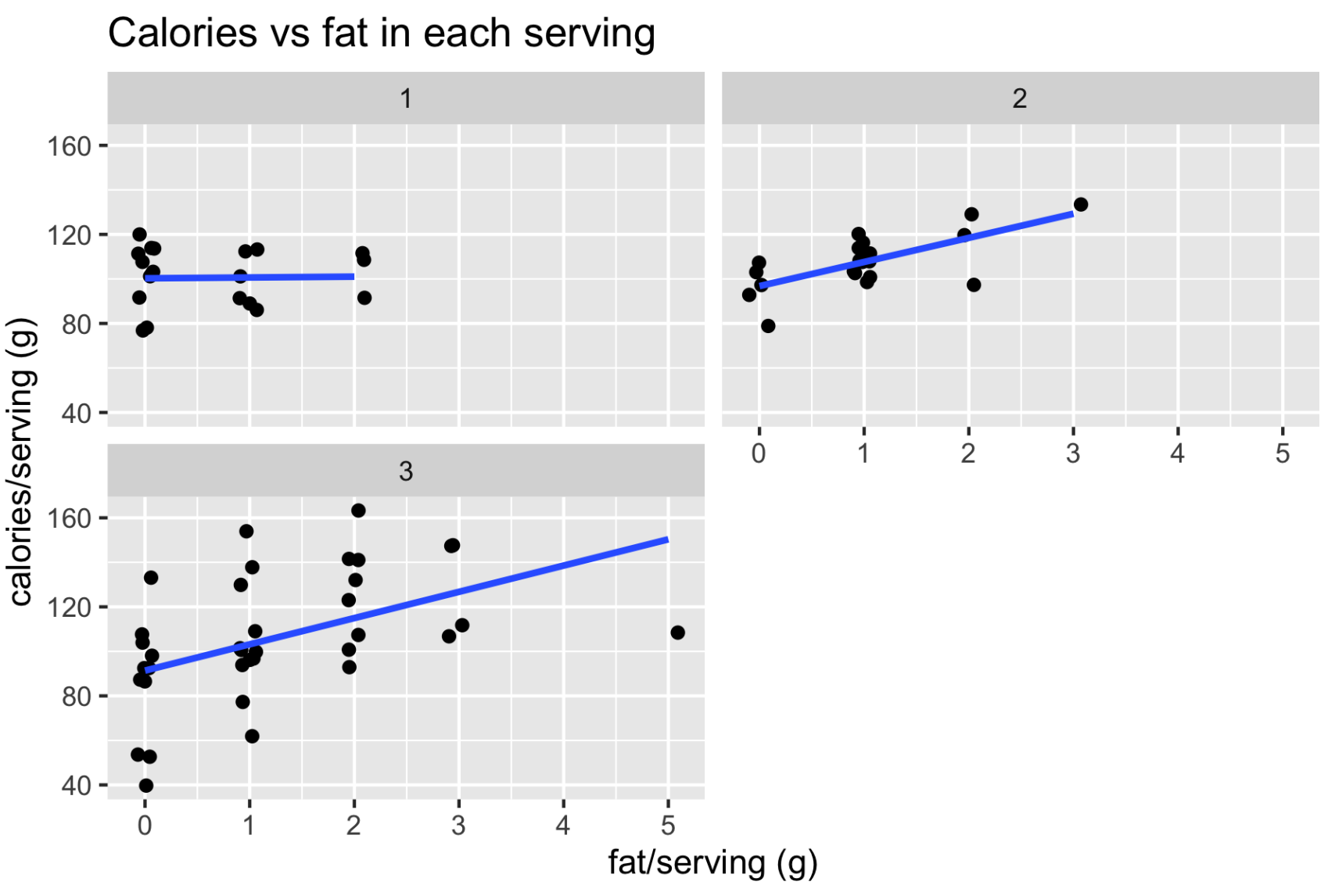
```
1 mitesfacet <- ggplot(data=butts, aes(x=Butts.weight, y=log.m
2     ggtitle("Mean number of mites by butts weight")+
3     xlab("Butts weight (g)")+ylab("log(number of mites)")+
4     geom_point()+
5     geom_smooth(method="loess", se=FALSE)+
6     facet_wrap( ~Species + Nest.content, ncol=3,
7         labeller=label_both)
8 mitesfacet
```

Plotting with *ggplot2* - Facetting - Exercise



Plotting with *ggplot2* - Facetting - Modifying facet labels

How to change facet values, i.e. 1="low", 2="medium", 3="high"

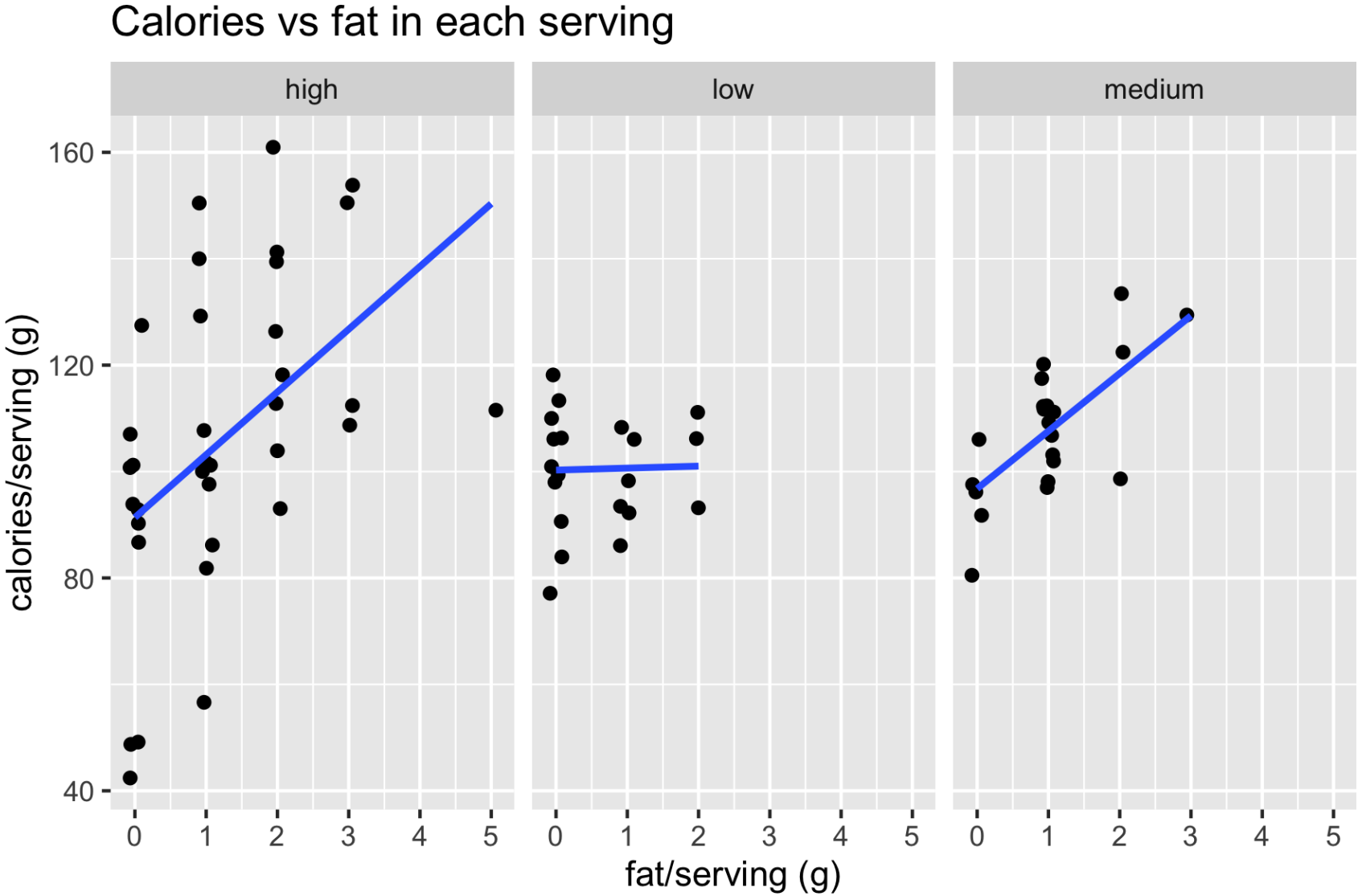


Three methods (different complexity)
(1) Use *recode()* to make new variable and use it.

- But ... order of facets may change.

```
cereal$shelf2 <- car::recode(cereal$shelf,
    "  1='low';
       2='medium';
       3='high'")
cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))+
     ggtitle("Calories vs fat in each serving")+
     xlab("fat/serving (g)")+ylab("calories/serving (g)")+
     geom_point(position=position_jitter(width=0.1))+
     geom_smooth(method="lm", se=FALSE)+
     facet_wrap(~shelf2, ncol=3)
cerealplot
```

Changing facet labels using *recode*

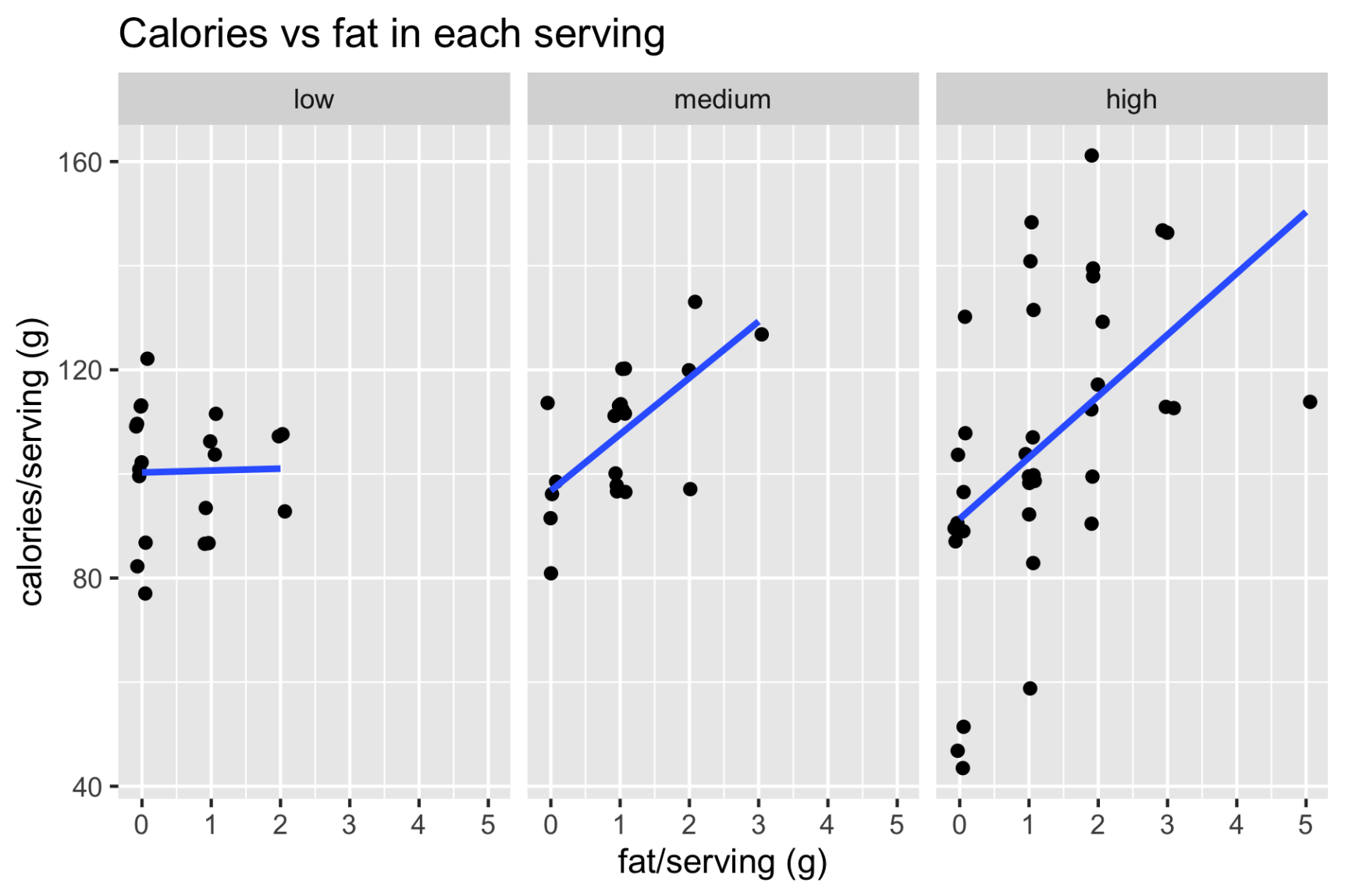


Three methods (different complexity)

(2) Create appropriate factor labels

```
1  cereal$shelfF2 <- factor(cereal$shelf, labels=c("low","mediu
2
3  cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))+
4       ggtitle("Calories vs fat in each serving")+
5       xlab("fat/serving (g)")+ylab("calories/serving (g)")+
6       geom_point(position=position_jitter(width=0.1))+
7       geom_smooth(method="lm", se=FALSE)+
8       facet_wrap(~shelfF2, ncol=3)
9  cerealplot
```

Changing facet labels using *factor*



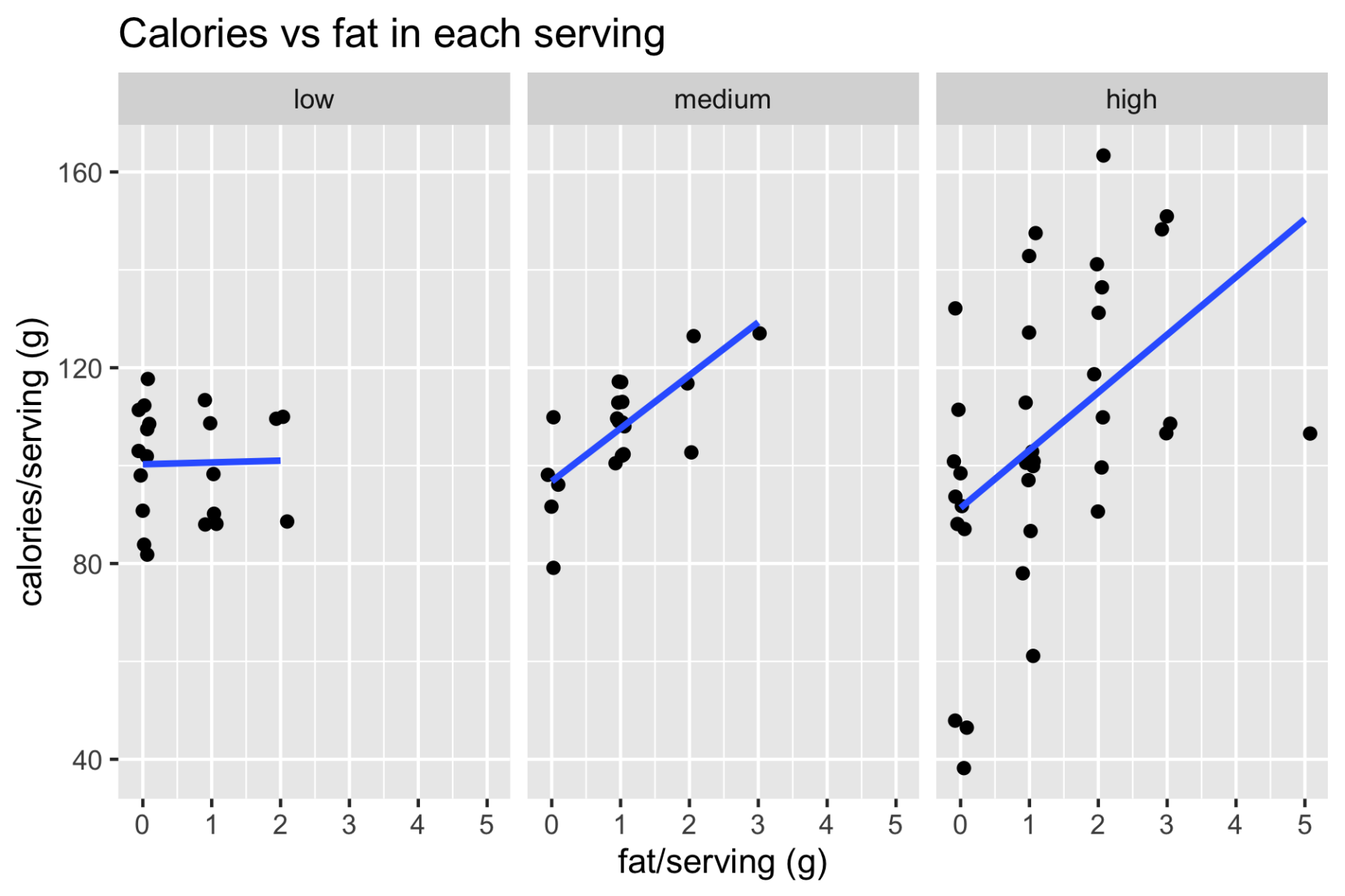
Three methods (different complexity)

(3) Use a labeller function

```
cereal$shelfc <- as.character(cereal$shelf)
shelf_names <- c('1' = "low",
                 '2' = "medium",
                 '3' = "high")
cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))+
     ggtitle("Calories vs fat in each serving")+
     xlab("fat/serving (g)")+ylab("calories/serving (g)")+
     geom_point(position=position_jitter(width=0.1))+
     geom_smooth(method="lm", se=FALSE)+
     facet_wrap(~shelfc, ncol=3,
        labeller=labeller(shelfc=as_labeller(shelf_names)))
cerealplot
```

Faceting variable MUST be character.

Changing facet labels using *labeller* argument.



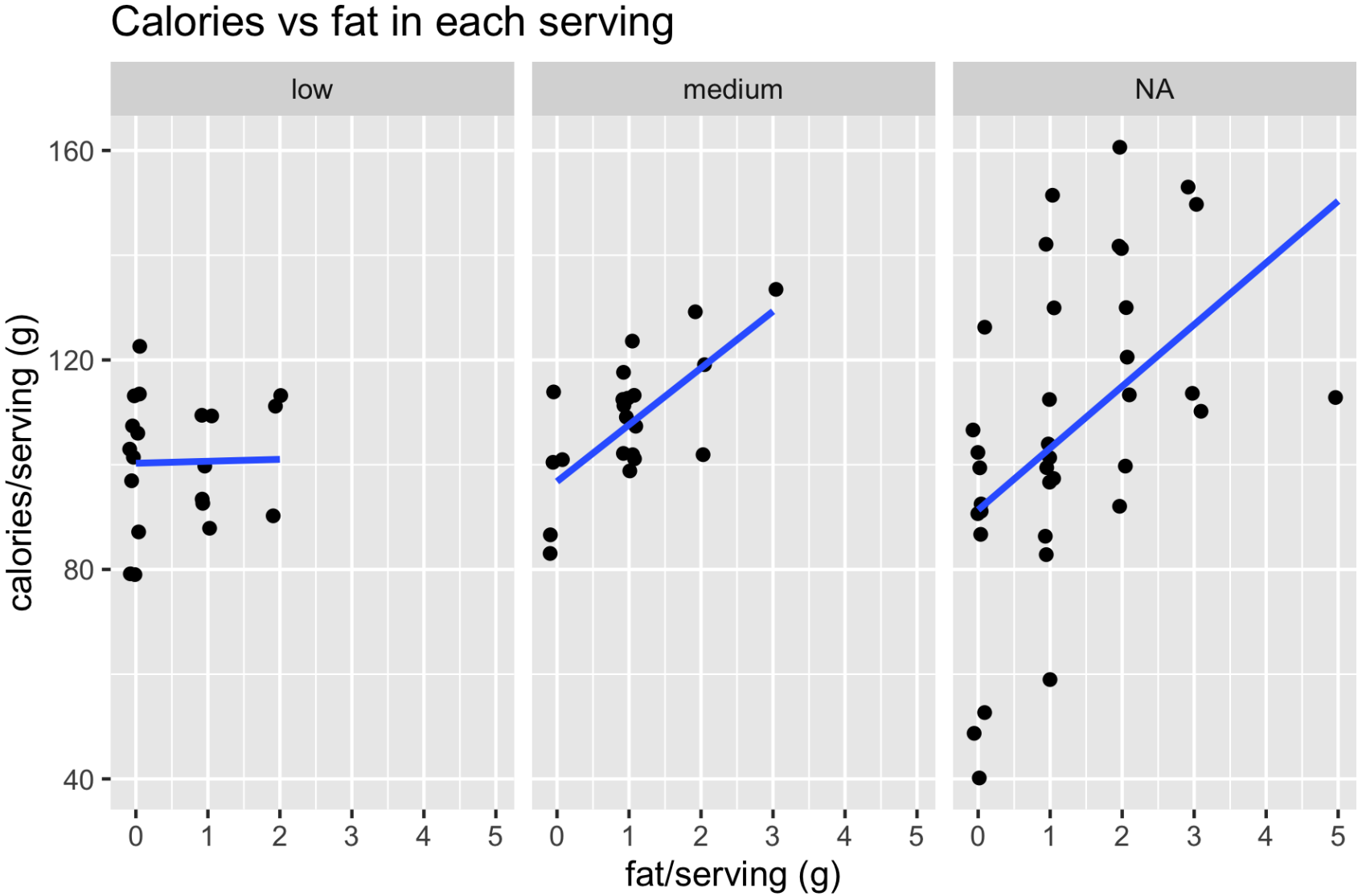
Three methods (different complexity)

(3) Use a labeller function

```
# What happens if there is mismatch?
cereal$shelfc <- as.character(cereal$shelf)
shelf_names <- c('1' = "low",
                 '2' = "medium",
                 '4' = "high")
```

Notice there is no shelf code 4.

Changing facet labels using *labeller* - impact of mismatch.

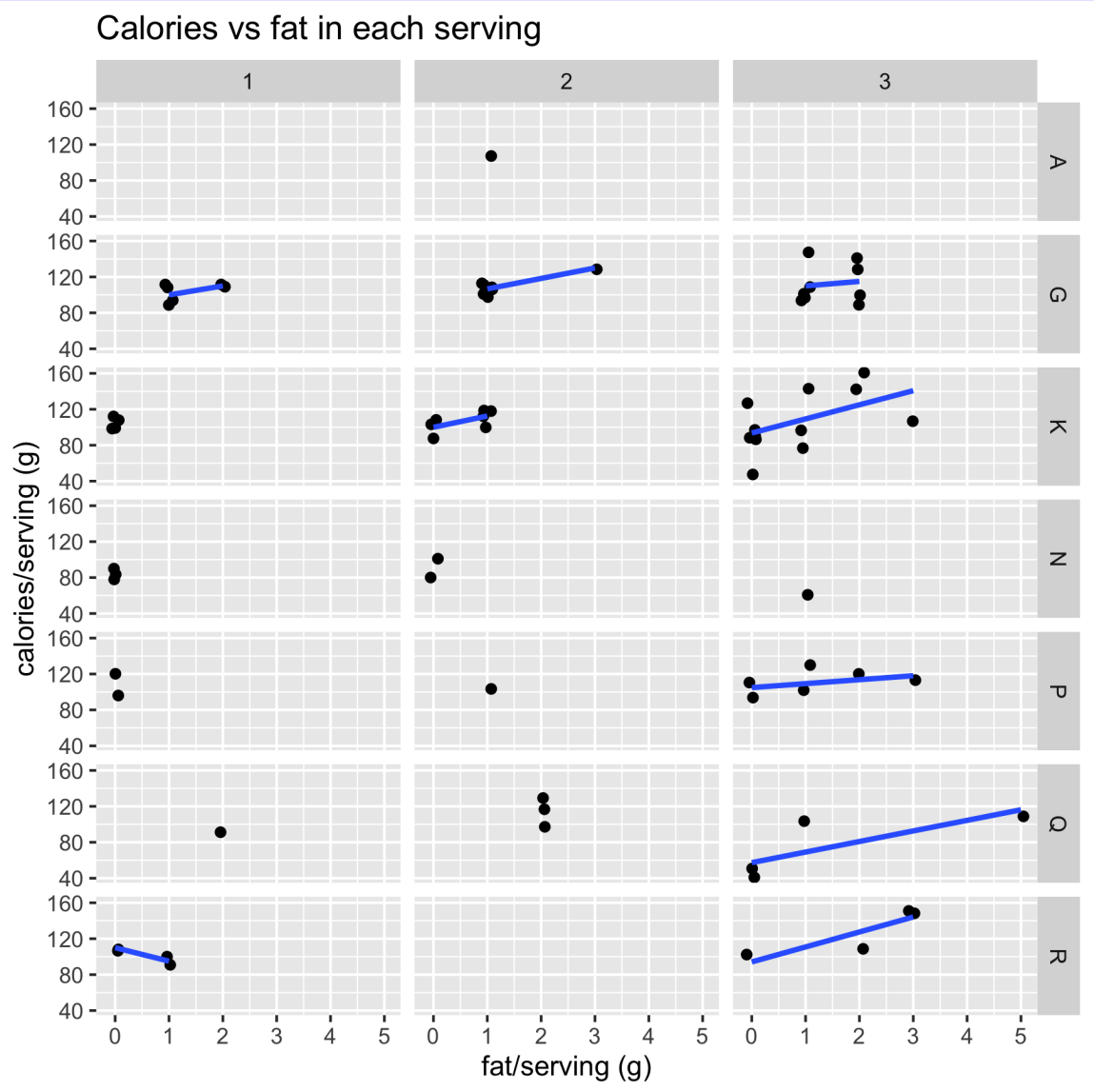


Plotting with *ggplot2* - Facetting - Multiple Pages

All facets must fit on a single page (groan) and may be too small.

```
cereal$shelfc <- as.character(cereal$shelf)
cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))+
     ggtitle("Calories vs fat in each serving")+
     xlab("fat/serving (g)")+ylab("calories/serving (g)")+
     geom_point(position=position_jitter(width=0.1))+
     geom_smooth(method="lm", se=FALSE)+
     facet_grid(mfr~shelfc)
cerealplot
```

Plotting with *ggplot2* - Facetting - Multiple pages



Plotting with *ggplot2* - Facetting - Multiple Pages

The *ggforce* package has two useful functions:

- *facet_grid_paginate(facets, nrow=, ncol=, page=)*
- *facet_wrap_paginate(facets, nrow=, ncol=, page=)*

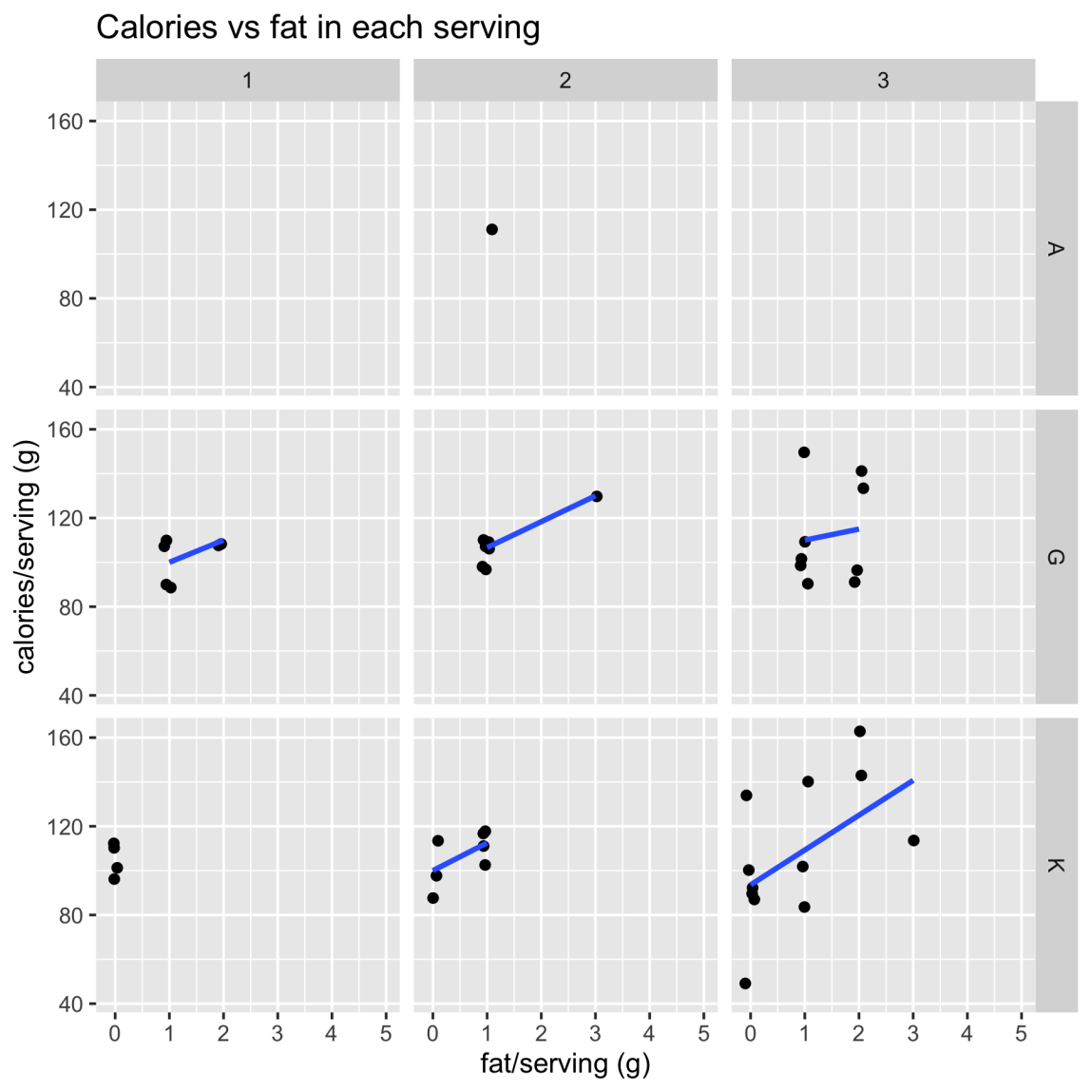
Proceed by example:

Plotting with *ggplot2* - Facetting - Multiple Pages

(1) Set up plot and view one or more of the pages

```
1  library(ggforce)
2  page=1
3  cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))+
4       ggtitle("Calories vs fat in each serving")+
5       xlab("fat/serving (g)")+ylab("calories/serving (g)")+
6       geom_point(position=position_jitter(width=0.1))+
7       geom_smooth(method="lm", se=FALSE)+
8       facet_grid_paginate(mfr~shelfc, nrow=3, ncol=3,
9          page=page)
10 cerealplot
```

Plotting with *ggplot2* - Facetting - Multiple pages



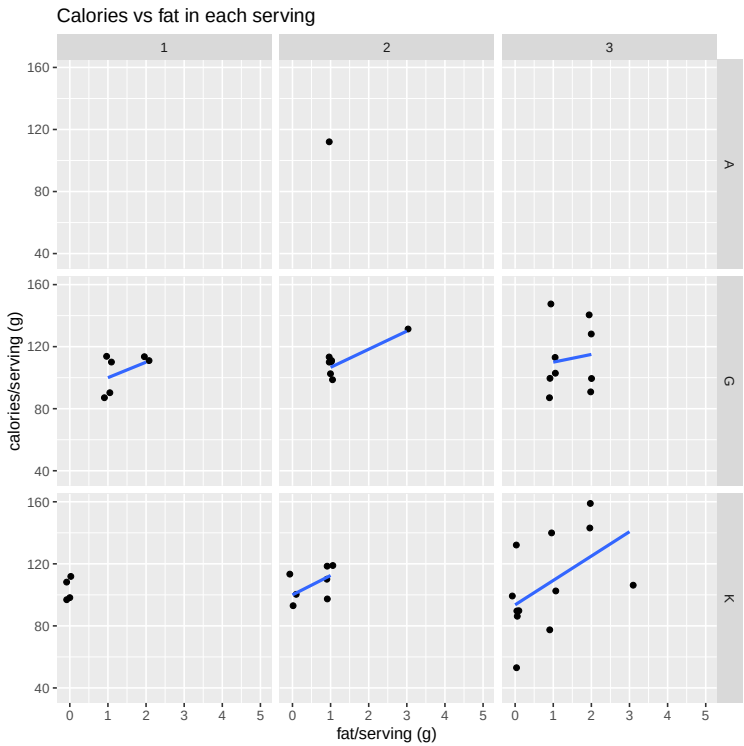
Try it with *page=2* etc.

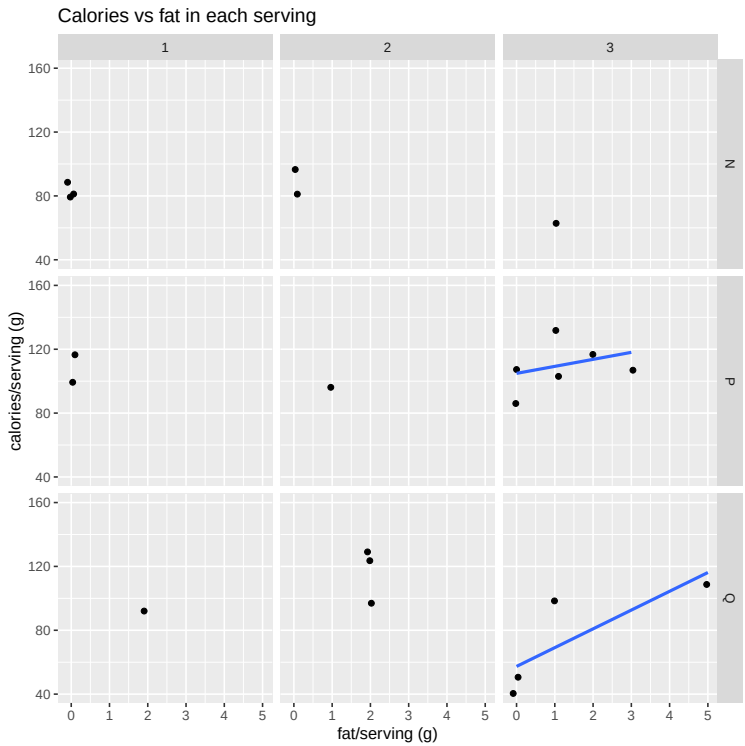
Plotting with *ggplot2* - Facetting - Multiple Pages

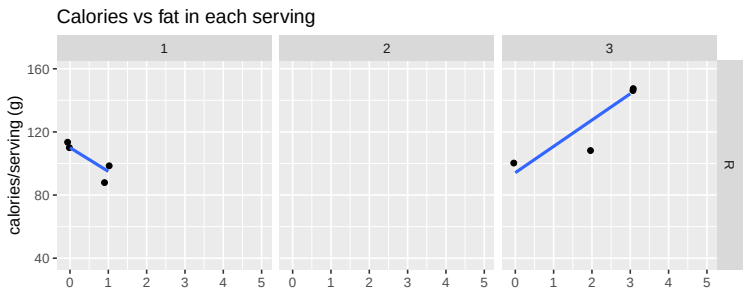
(2) Save the multi-page plot.
Many graphic formats (e.g. **.png* do NOT allow multiple pages) so may have to wrap in to a *pdf* file

```
1  library(ggforce)
2  pdf(file=".....")
3  plyr::l_ply(1:3, function(page){
4     cerealplot <- ggplot(data=cereal, aes(x=fat, y=calories))
5       ggtitle("Calories vs fat in each serving")+
6       xlab("fat/serving (g)")+ylab("calories/serving (g)")+
7       geom_point(position=position_jitter(width=0.1))+
8       geom_smooth(method="lm", se=FALSE)+
9       facet_grid_paginate(mfr~shelfc, nrow=3, ncol=3,
10         page=page)
11    plot(cerealplot) # must be explicit in loops
12 })
13 dev.off() # don't forget the dev.off() to close the file.
```

Don't forget the *dev.off()* to close the **.pdf* file. Unfortunately, you need to figure out the number of pages (the *n_pages()* has a bug.

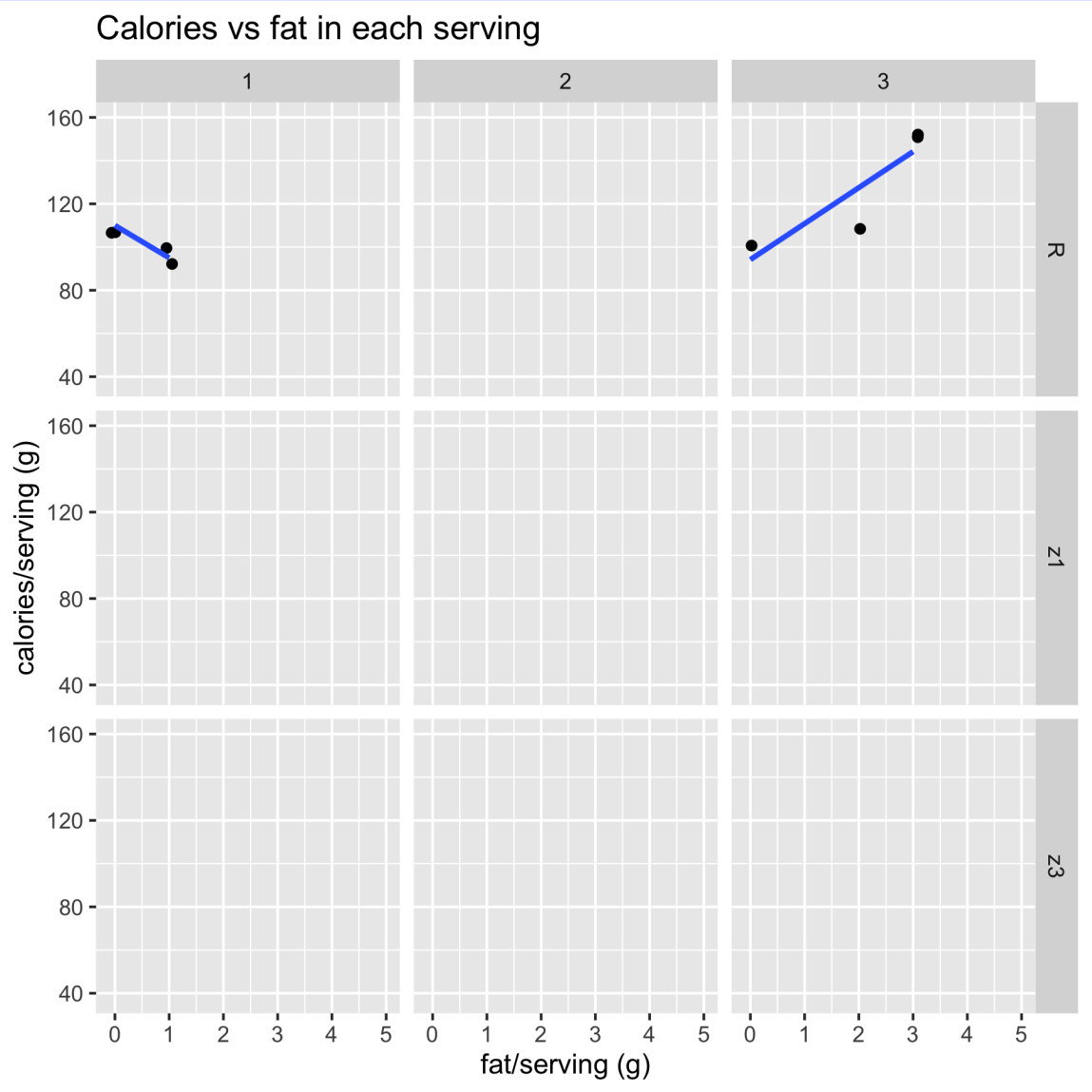






fat/serving (g)

Plotting with *ggplot2* - Facetting - Multiple pages



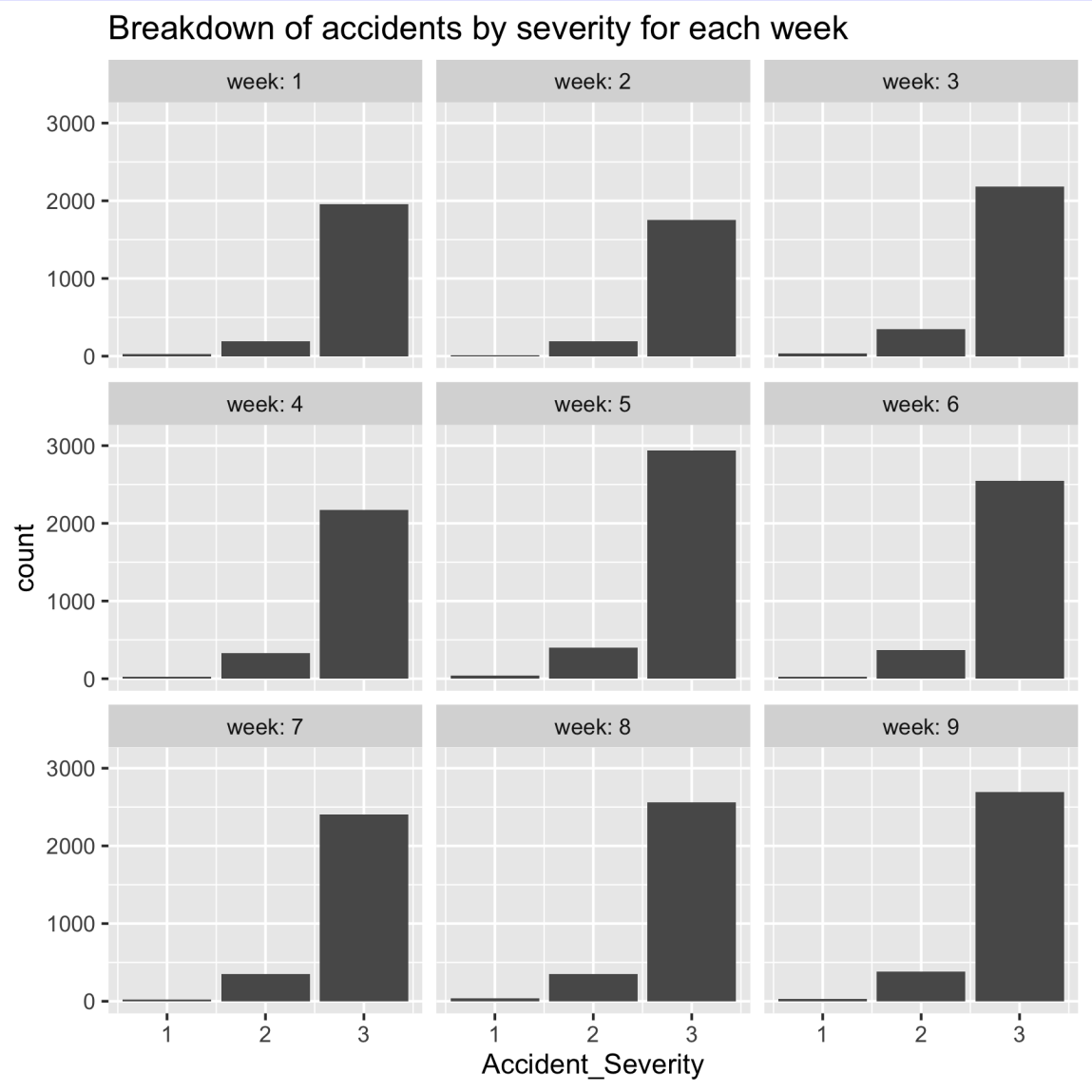
Refer to the *accidents* dataset.

- Make a bar chart of the accident severity by week (for all 52 weeks)
- Use a labeller function to label the facets

Do you notice anything unusual?
Hints:

- the *lubridate::week(...date...)* can generate the week in the year

Plotting with *ggplot2* - Facetting - Exercise



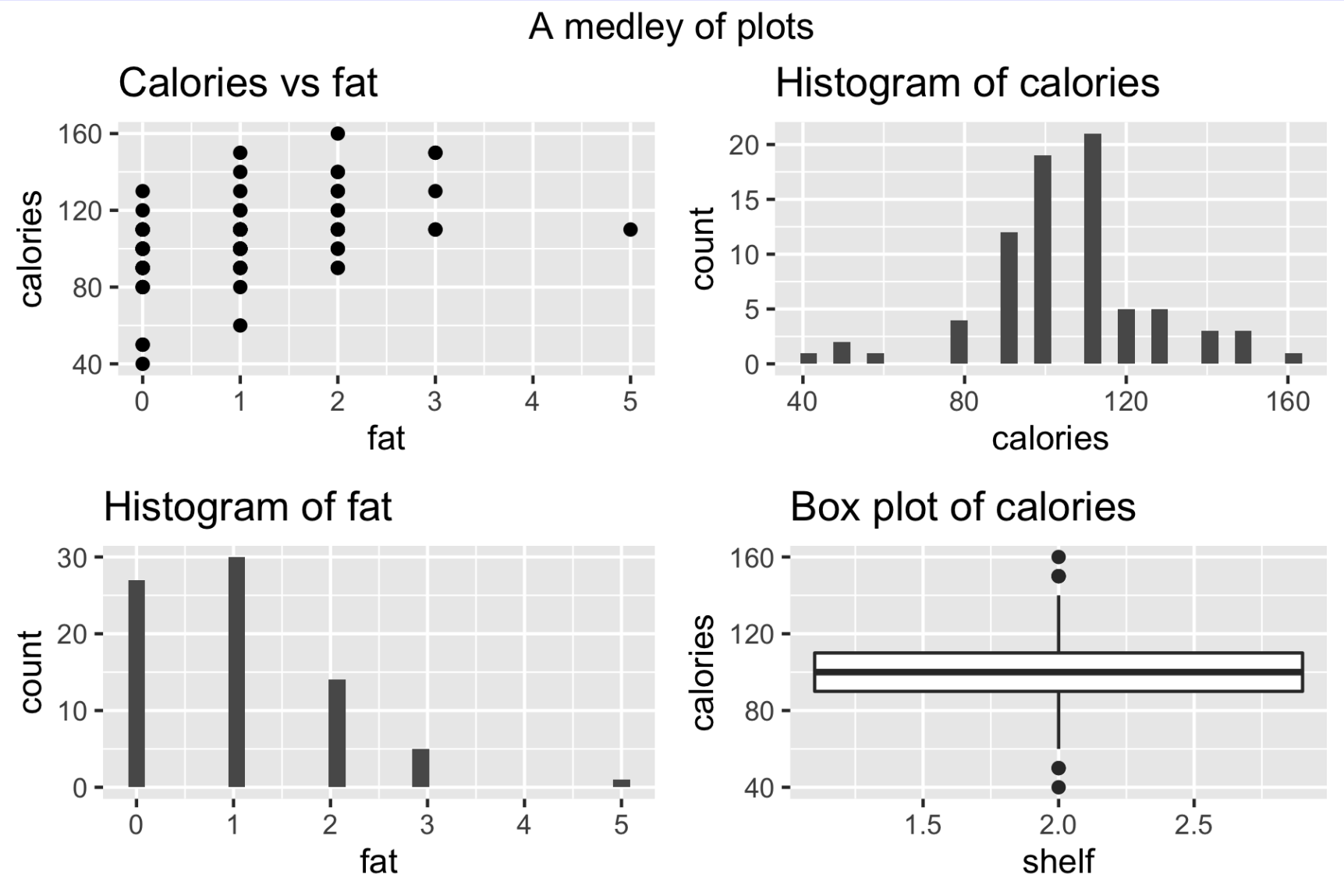
Multiple plot types per page

Plotting with *ggplot2* - Multiple different plots/page

```
1  library(gridExtra)
2  p1 <- ggplot(data=cereal, aes(x=fat, y=calories))+
3        ggtitle("Calories vs fat")+ geom_point()
4
5  p2 <- ggplot(data=cereal, aes(x=Calories))+
6        ggtitle("Histogram of Calories")+geom_histogram()
7
8  p3 <- ggplot(data=cereal, aes(x=fat))+
9        ggtitle("Histogram of Fat")+ geom_histogram()
10
11 p4 <- ggplot(data=cereal, aes(x=shelfF, y=calories))+
12       ggtitle("Box plot of calories")+geom_boxplot()
13
14 allplot <- arrangeGrob(p1,p2,p3,p4, nrow=2,
15              top="A medley of plots")
16 plot(allplot)
```

Refer to vignettes in *gridExtra* package.

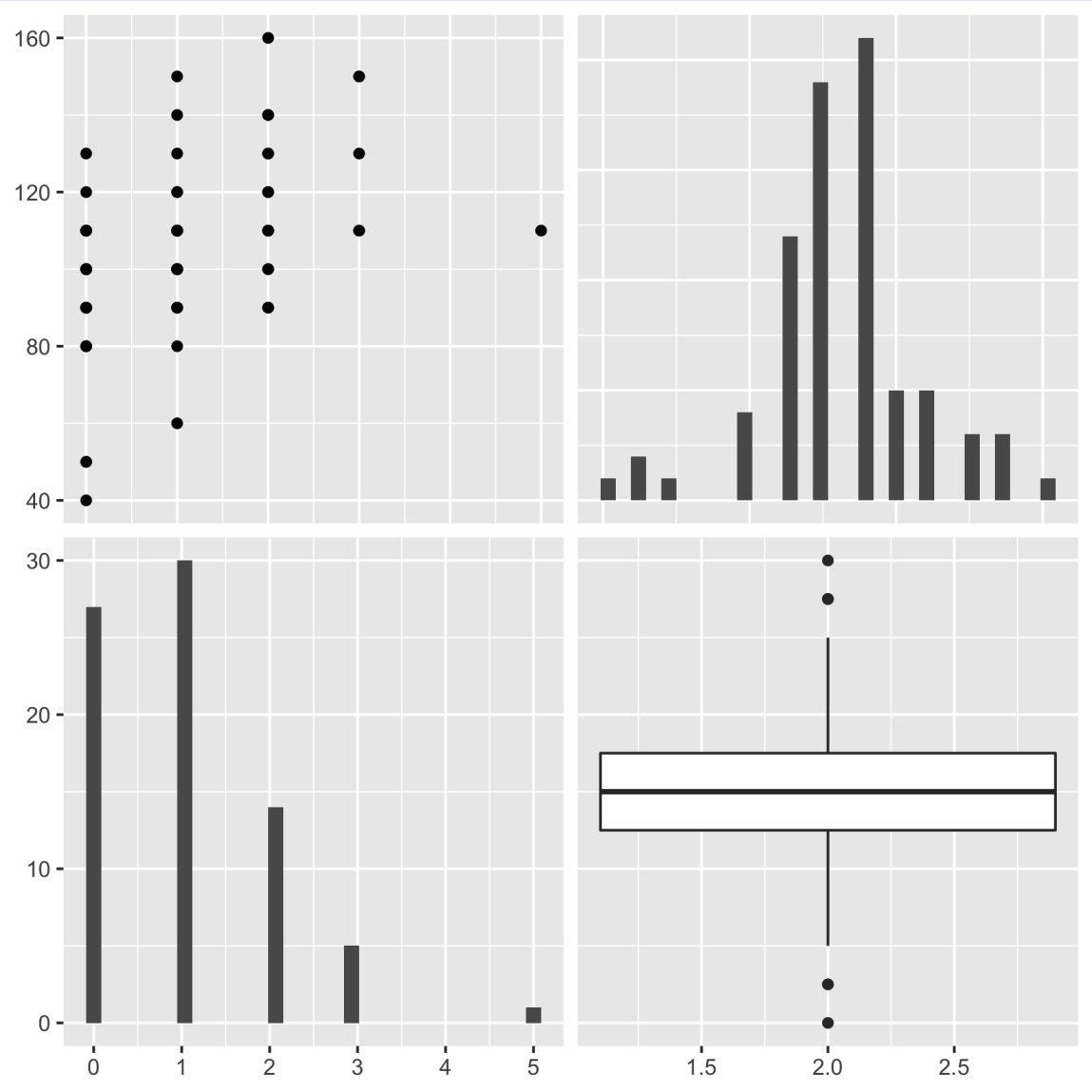
Plotting with *ggplot2* - Multiple-plots/page



Plotting with *ggplot2* - Multiple different plots/page

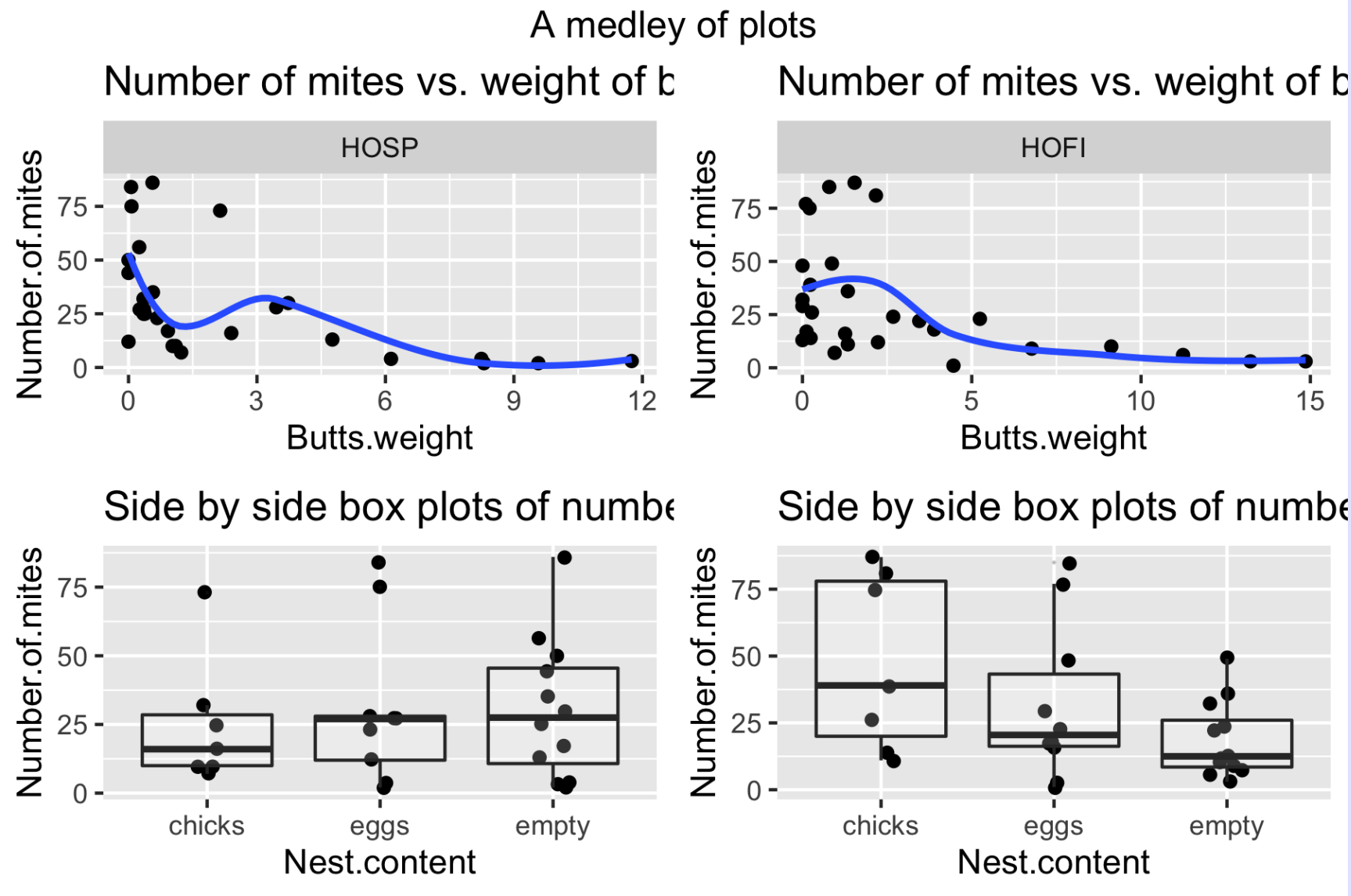
```
1  library(GGally)
2  plot.list <- list(p1, p2, p3, p4)
3  allplot <- GGally::ggmatrix(plot.list,
4                              ncol=2, nrow=2)
5  allplot
```

Plotting with *ggplot2* - Multiple-plots/page



Plotting with *ggplot2* - Multiple-plots/page - Exercise

Produce the following plot from the Birds 'n Butts data;

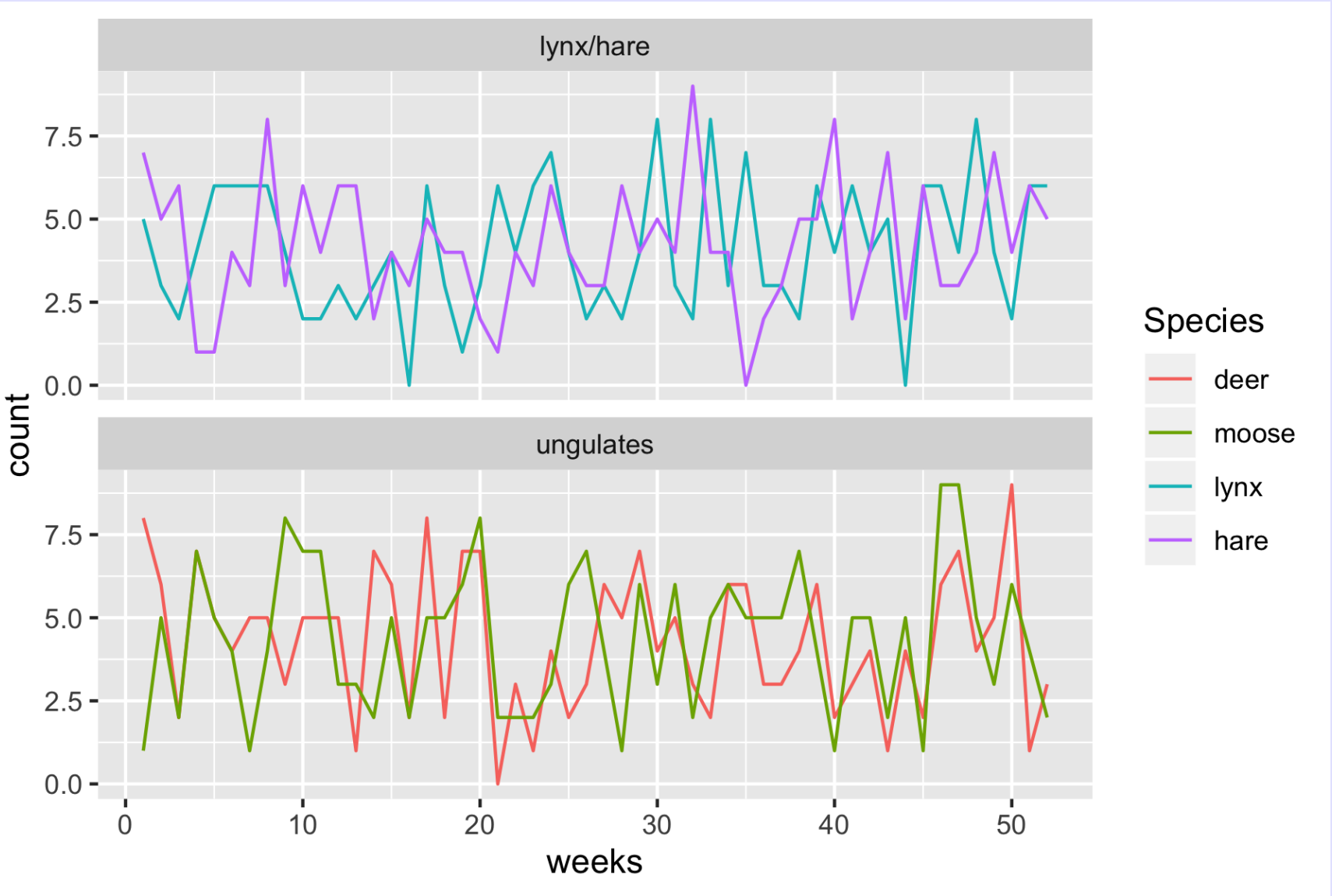


Different legends on each facet

Combination faceting and *arrangeGrob()*

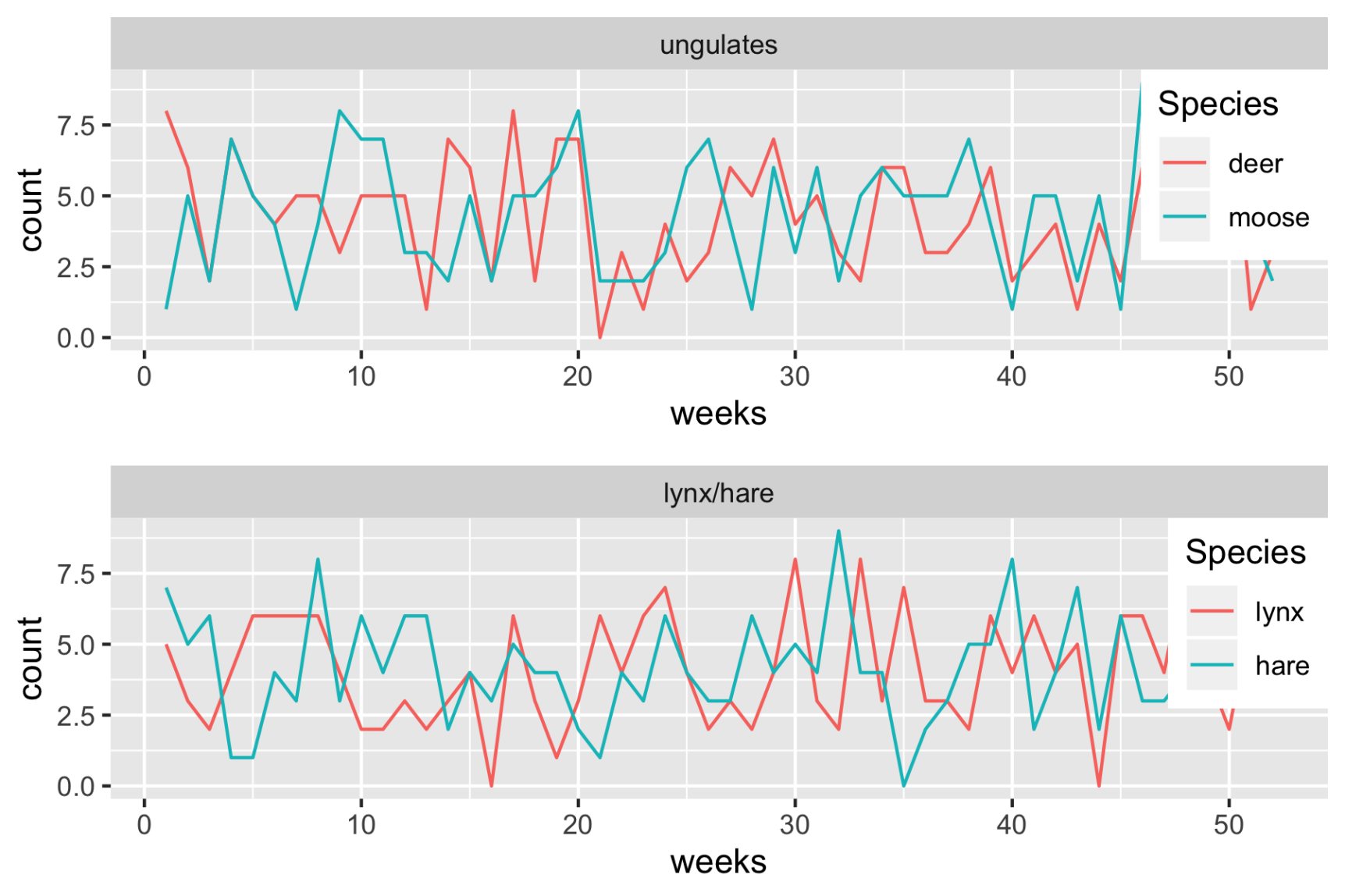
Plotting with *ggplot2* - Different legend on each plot

We want to change from:



Plotting with *ggplot2* - Different legend on each plot

We want to change to:



Plotting with *ggplot2* - Different legend on each plot

Short answer is NO:

`https://stackoverflow.com/questions/34956350/ggplot-different-legends-for-different-facets`

> *The design philosophy behind facets are that they are sub-plots that share aesthetics, which is not what you want.*

Plotting with *ggplot2* - Different legend on each plot I

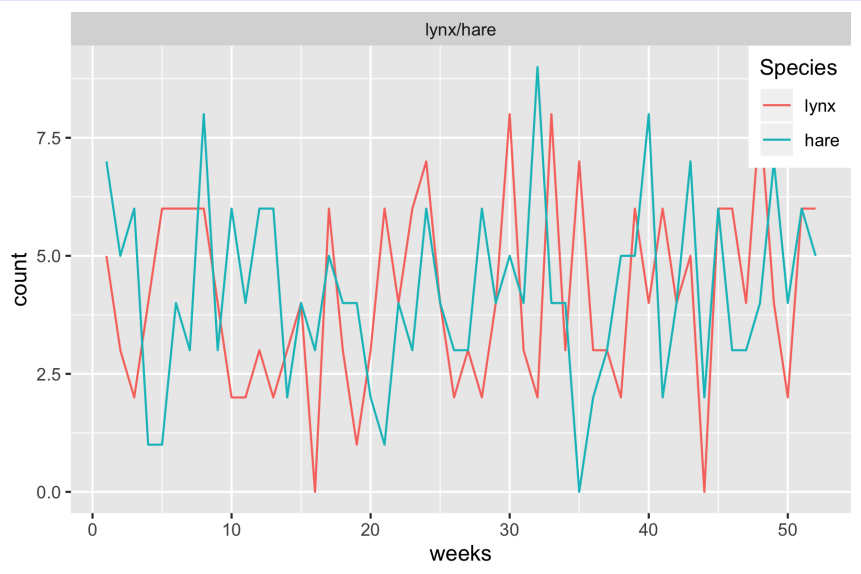
But follow the trick at
https://stackoverflow.com/questions/14840542/place-a-legend-for-each-facet-wrap-grid-in-ggplot2

Part 1: Generate TWO separate plots with their own legend

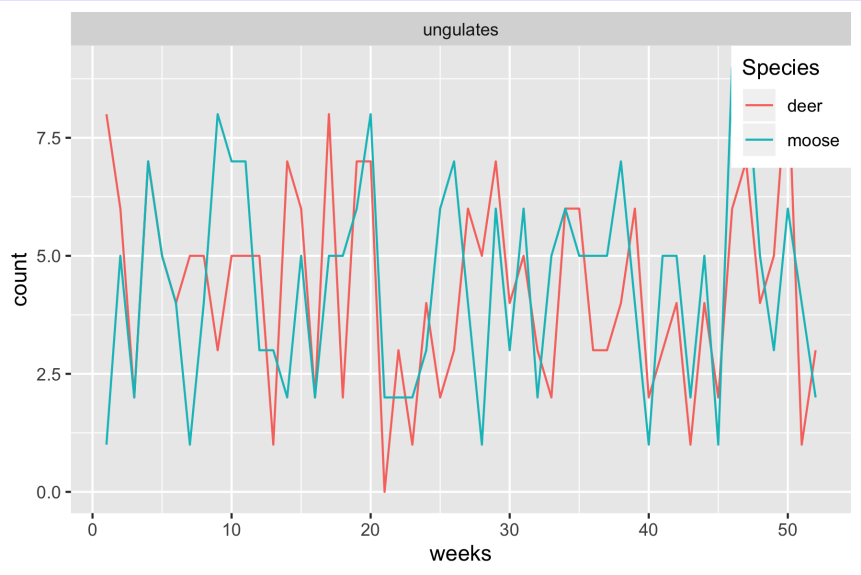
```
ggList <- plyr::dlply(counts,"Group", function(groupdata) {
  ggplot(data=groupdata,
         aes(x=weeks, y=count, colour = Species)) +
    geom_line() +
    facet_wrap(~Group, ncol=1)+
    theme(legend.position=c(1,1),
          legend.justification=c(1,1))
  })
ggList
```

Plotting with *ggplot2* - Different legend on each plot II

This gives:



Plotting with *ggplot2* - Different legend on each plot III



Plotting with *ggplot2* - Different legend on each plot IV

Part 2: Stack the two separate plots using *arrangeGrob*

```
1 allplot <- do.call(arrangeGrob, ggList)
2 plot(allplot)
```

Check my *Rcode* for more details

Plot on a map

Brief intro - refer to spatial data section for more details

Package *ggmap* - map data

```
1  library(ggmap)
2  sfu.coord <- c(-122.917957, 49.276765 )
3
4  my.drive.csv <- textConnection("
5  long, lat
6  -122.84378900000002, 49.29009199999999
7  -122.82799615332033, 49.28426960031931
8  -122.82696618505861, 49.27755059244836
9  -122.86679162451173, 49.2767664856581
10 -122.88790597387697, 49.26276555269492
11 -122.90833367773439, 49.2653420526345l
12 -122.92532815405275, 49.273518748310764
13 -122.91434182592775, 49.277662583414 39")
14 my.drive <- read.csv(my.drive.csv, header=TRUE, as.is=TRUE,
```

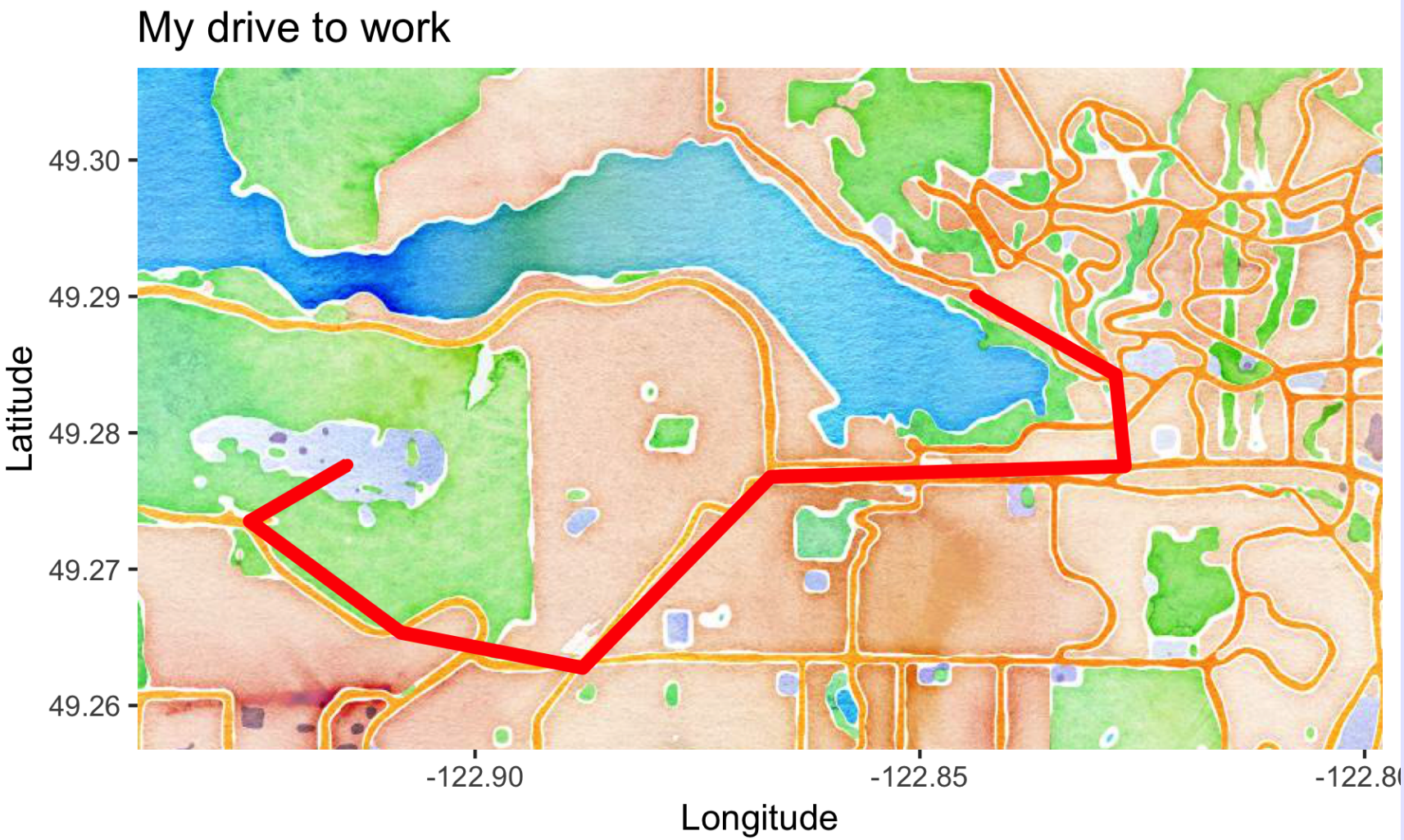
ggmap - map data

```
1 my.map.dl <- ggmap::get_map(c(left=sfu.coord[1]-.02, bottom=
2                 maptype="watercolor",  source="stamen")
3 my.map <- ggmap(my.map.dl)
4
5 plot1 <- my.map +
6          ggtitle("My drive to work")+
7          geom_point(data=my.drive, aes(x=long, y=lat),size=1
8          geom_path(data=my.drive, aes(x=long, y=lat), color=
9          ylab("Latitude")+xlab("Longitude")
```

You now need to register to retrieve maps.

Note that you need to register to retrieve maps from Google.

Plotting map data

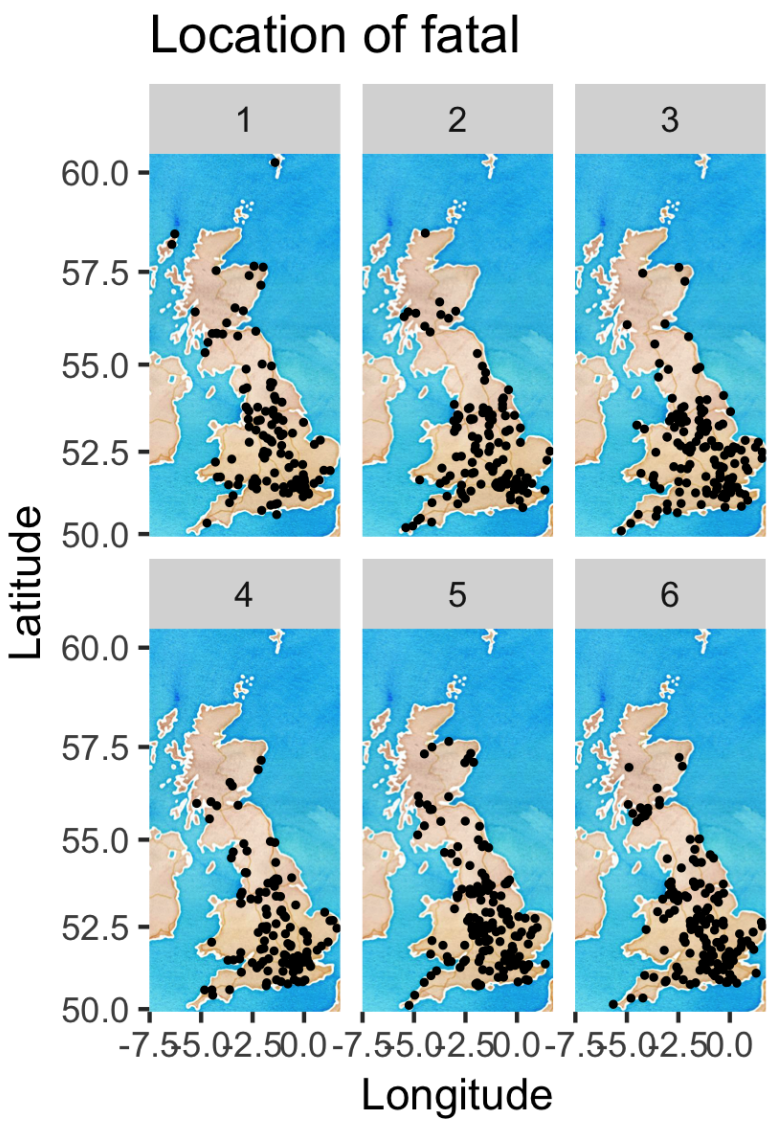


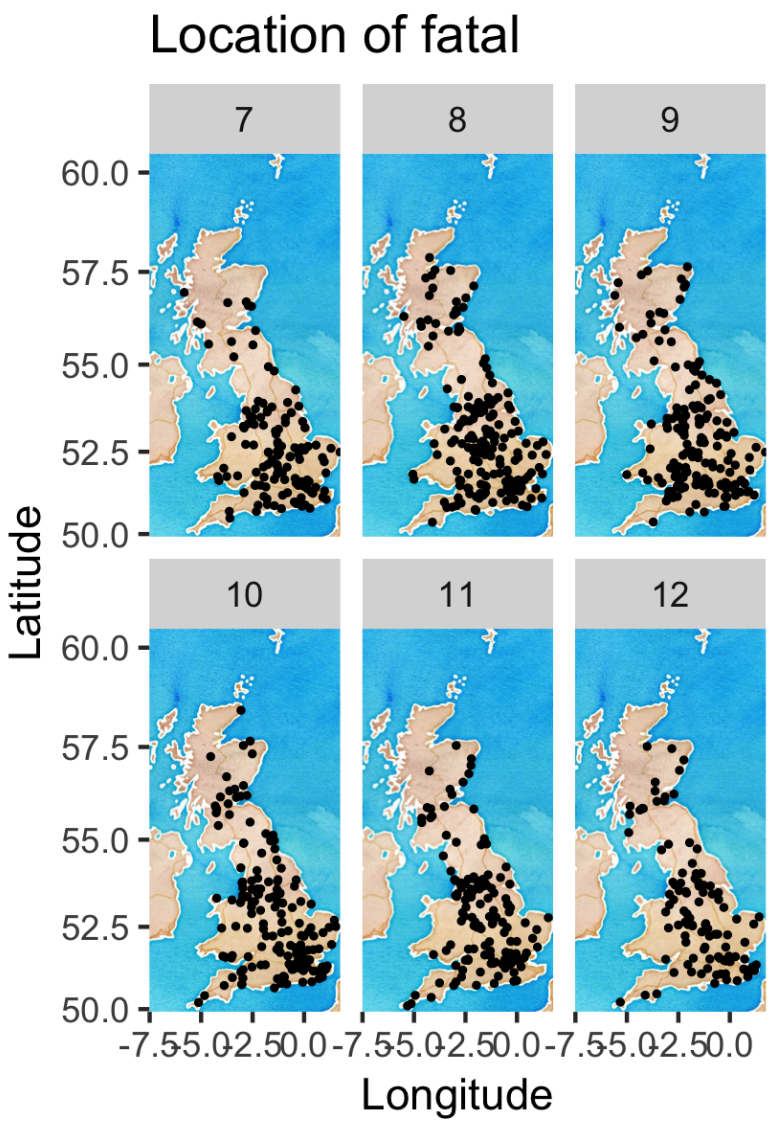
Plotting with *ggplot2* - ggmap - Exercise

Refer to accidents dataset:

- Extract the fatal accidents
- Plot the locations of the fatal accidents by month (Hint:: *lubridate::month()*

Plotting with *ggplot2* - ggmap - Exercise




Location of fatal
7
8
9
10
11
12
60.0
57.5
55.0
52.5
50.0
Latitude
-7.5
-5.0
-2.5
0.0
Longitude

Shiny - Perils of non-standard evaluation I

We often want to select VARIABLES to plot. Run the following:

```
1  # Consider the following plot
2  ggplot(data=cereal, aes(y=calories, x=fat))+
3    geom_point()
4
5  # Suppose we wish to "program" the variables using somethin
6  xvar <- 'fat'
7  yvar <- 'calories'
8
9  # this fails because of non-standard evaluation
10 ggplot(data=cereal, aes(y=yvar, x=xvar))+
11   geom_point()
```

What happens is a problem of non-standard evaluation that occurs throughout *R*!
What does *y=yvar* mean and how is it distinguished from *y=cereal*?

ggplot2 includes the *aes_string* to deal with this problem. *dplyr* also allows for non-standard evaluation.

```
1 xvar <- 'fat'
2 yvar <- 'calories'
3 ggplot(data=cereal, aes_string(y=yvar, x=xvar))+
4   geom_point()
```

This does what you want.
Similarly

```
1 # use [] in regular data frames
2 yvar <- 'calories'
3 cereal$yvar # returns null
4 cereal$"yvar" # also returns null
5 cereal[, yvar, drop=FALSE]
```

Non-standard evaluation - Exercise

Create a function that:

- Takes a data frame, name of X and Y variable and makes a scatterplot
- Add the regression line
- Add the fitted line to the plot.

Non-standard evaluation - Exercise I

```
1  # a function that plots two specified variables with the sm
2  # and add the regression equation to the graph
3  myplot <- function(in.data, xvar, yvar){
4     s.plot <- ggplot(data=in.data, aes_string(x=xvar, y=yvar)
5        ggtitle(paste("Plot of ", yvar, " vs. ", xvar, " with
6        geom_point( position=position_jitter(h=.1, w=.1))+
7        geom_smooth(method="lm", se=FALSE)
8     # now we have the reverse problem of non-standard evalua
9     # The following doesn't work
10    #     fit <- lm(yvar ~ xvar, dat=in.data)
11    # Here is one method. There are many other ways
12    # See http://adv-r.had.co.nz/Computing-on-the-language.h
13    fit <- lm( in.data[,yvar] ~ in.data[,xvar])
14    eqn <- paste(yvar, ' =', coef(fit)[1], " + (",coef(fit)[2
15    s.plot <- s.plot +
16       annotate("text", label=eqn, x=-Inf, y=Inf, hjust=0, v
17    s.plot
18 }
```

Non-standard evaluation - Exercise II

```
19
20  myplot(cereal, "fat", "calories")
```

Other useful add ons to ggplot()

Plotting with *ggplot2* - Additional packages

Visit `http://www.ggplot2-exts.org/gallery/` for more packages to add to *ggplot2* functionality.
Here are a few common ones that I frequently use.

Plotting with *ggplot2* - Additional packages

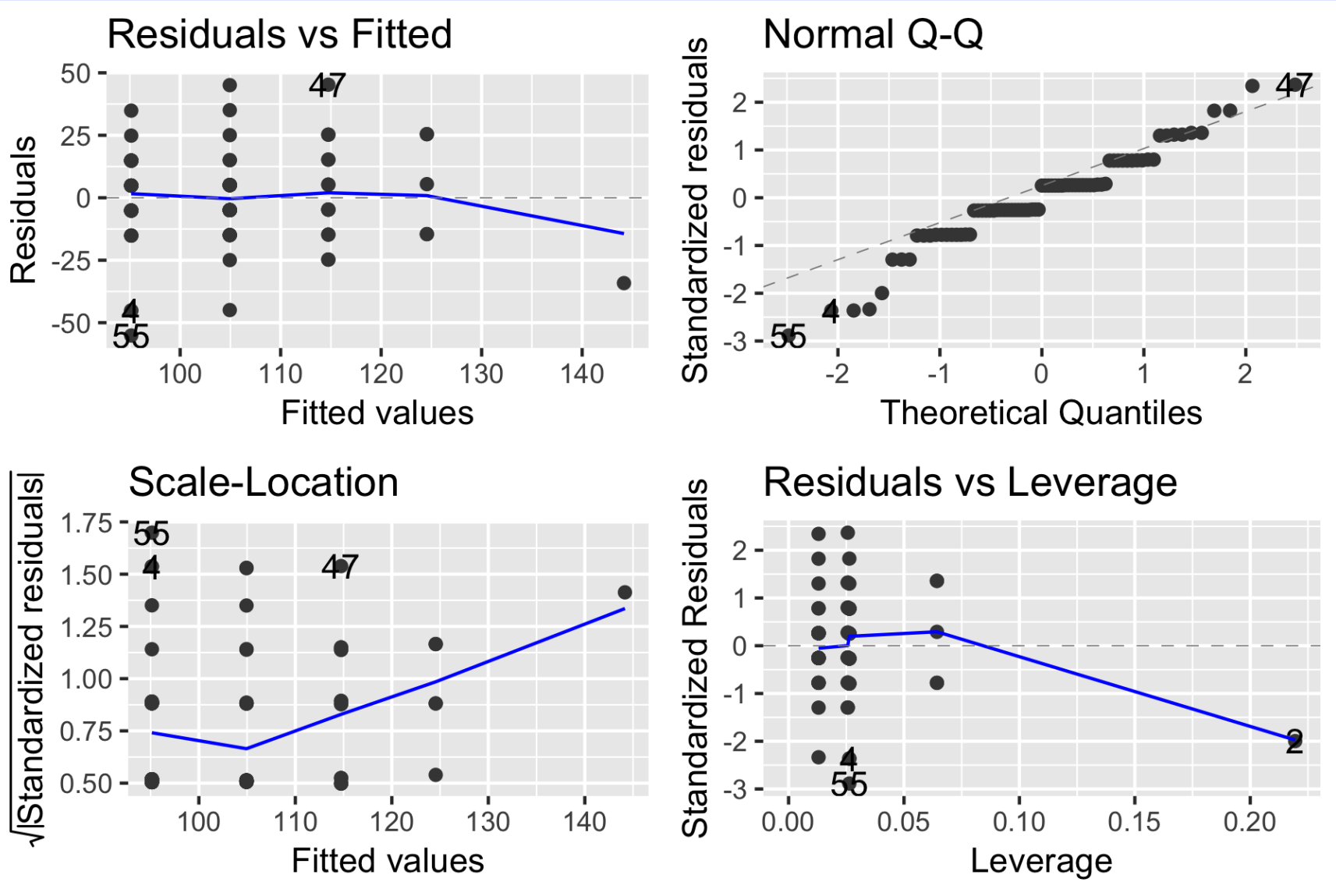
ggfortify - for diagnostic plots after model fits in place of *plot()*

```
1 library(ggfortify)
2 lm.fit <- lm(calories ~ fat, data=cereal)
3 diagplot <- autoplot(lm.fit)
4 diagplot
```

Plotting with *ggplot2* - additional packages

Diagnostic plots from *lm()* and *glm*

Additional functions available in *schwarz.functions.r*



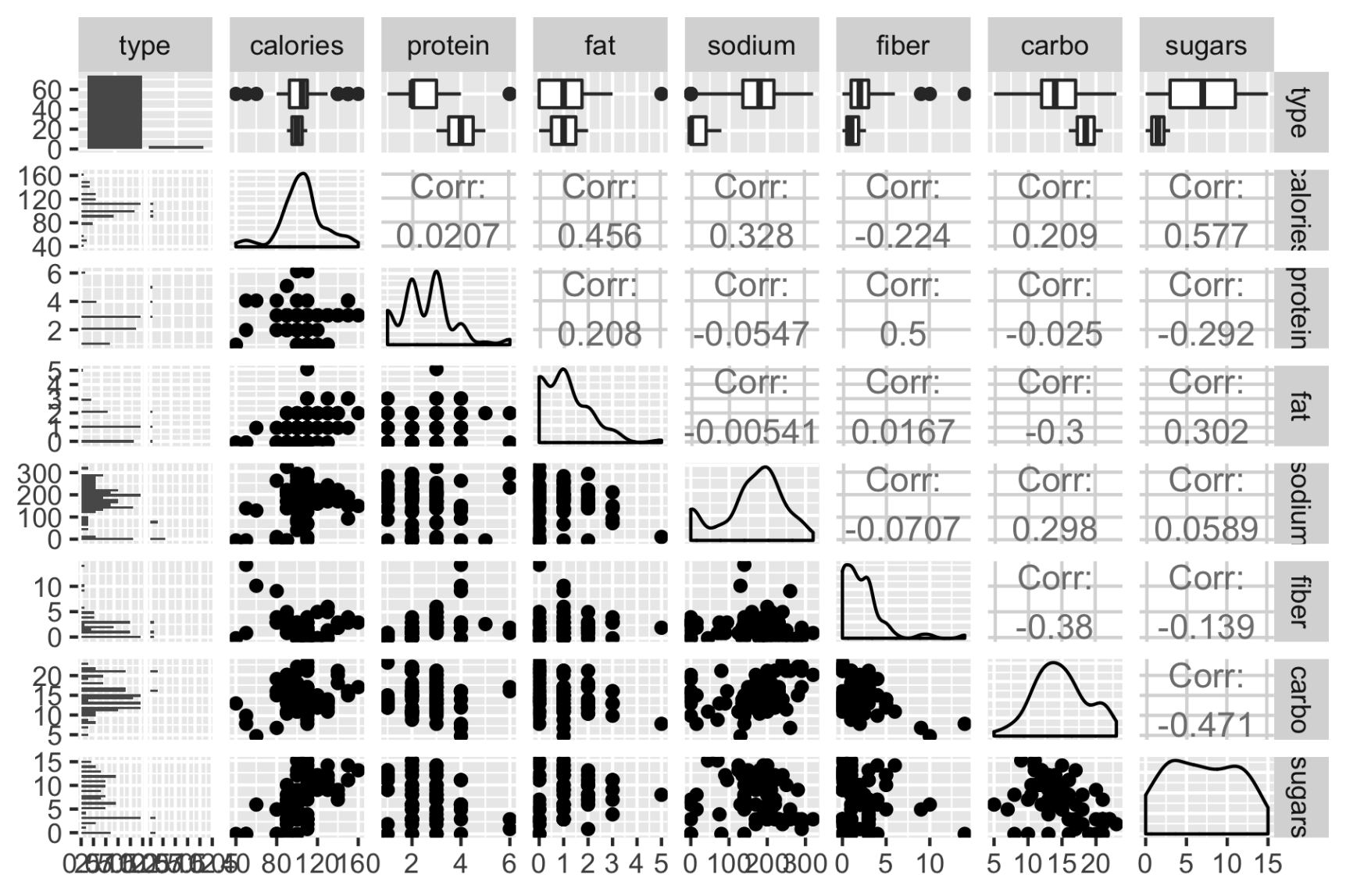
Plotting with *ggplot2* - Additional packages

GGally - for scatterplot matrix and grouping multiple plots together (see later)

```
1 library(GGally)
2 spm <- ggpairs(cereal, col=c(3,4,5,6,7,8,9,10))
3 spm
```

Plotting with *ggplot2* - additional packages

Scatterplot matrix



ggplot() Advanced Plotting - Summary

Avoid Base *R* graphics and use *ggplot()*!

- *facet_grid()* and *facet_wrap()* - very useful in clever ways
- *facet_grid_paginate()* useful for large sets of plots
- Many other packages with add ons