Basic Data Reduction with dplyr

John D. Lee and Linda Ng Boyle
10/1/2018

Google Quantitative User Experience Research

https://careers.google.com/jobs#!t=jo&jid=/google/quantitative-user-experience-researcher-google-building-41-1600-amphitheatre-1285660001&

"Develop code and statistical models to understand user experience."

"Experience in a programming language commonly used for data manipulation and computational statistics (such as Python, R, Matlab, C++, Java or Go), and with SQL."

`dplyr` is a simple way to do SQL-type data manipulation

dplyr for Data Reduction

dplyr as grammar of data reduction

- `select` to remove variables (i.e., columns)
- `filter` to subset the data and remove observations (i.e., rows)
- `mutate` to create new variables
- `summarise` to aggregate data across rows (i.e., mean of values)
- `group_by` to group variables for `mutate` and `summarise`
- `do` to fit a model or create a plot each group of the data
- `join` to combine datasets

Pipe (i.e., `%>%`) for Combining Operations

– `%>%` acts as "and then" to chain operations

– `sleep.df %>% summarise(mean(Reaction))` ~ take sleep dataframe "and then" calculate mean of Reaction

– More comprehensible than the equivalent: `summarise(sleep.df, mean(Reaction))`

– Many operations can be chained with `%>%` to create data reduction "sentences"

– `sleep.df %>% filter(Reaction>250) %>% group_by(Subject) %>% summarise(m.rt = mean(Reaction))`

More understandable?

– `summarise(group_by(filter(sleep.df, Reaction>250), Subject), mean(Reaction))`

`select` to Remove Variables

```
sleep.df = sleepstudy

head(sleep.df)
```

```
##   Reaction Days Subject
## 1 249.5600    0     308
## 2 258.7047    1     308
## 3 250.8006    2     308
## 4 321.4398    3     308
## 5 356.8519    4     308
## 6 414.6901    5     308
```

```
nort.sleep.df = sleep.df %>% select(-Reaction)
head(nort.sleep.df)
```

```
##   Days Subject
## 1    0     308
## 2    1     308
## 3    2     308
## 4    3     308
## 5    4     308
## 6    5     308
```

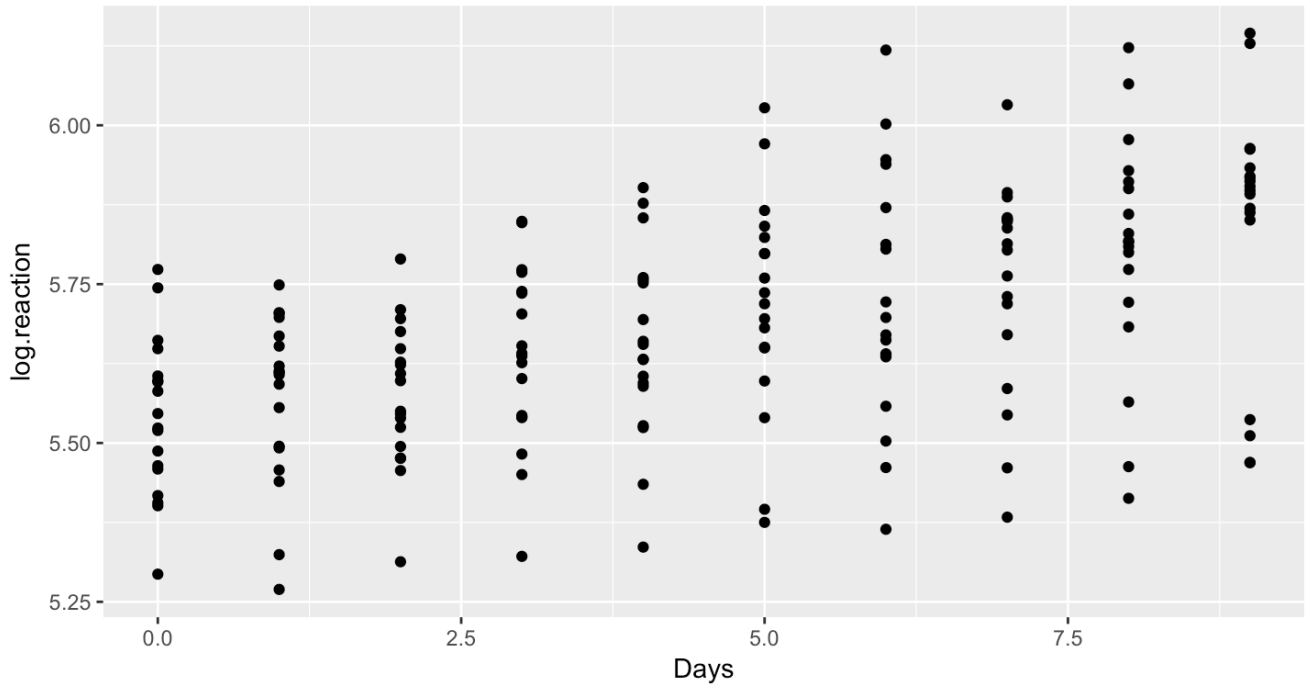
`filter` to Identify Outliers

```
longrt_sleep.df = sleep.df %>% filter(Reaction>450)

head(longrt_sleep.df)
```

```
##   Reaction Days Subject
## 1 466.3535    9     308
## 2 454.1619    6     332
## 3 455.8643    8     337
## 4 458.9167    9     337
```

`mutate` to Create New Variables

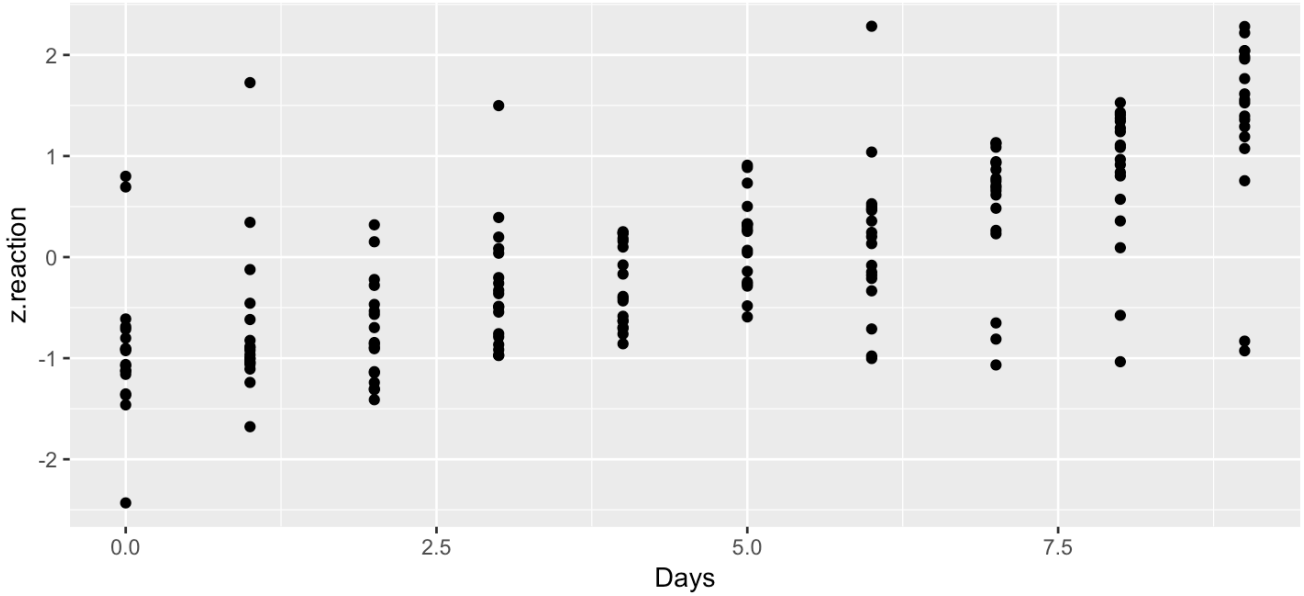
```
sleep.df = sleepstudy
sleep.df = sleep.df %>% mutate(log.reaction = log(Reaction))

ggplot(sleep.df, aes(Days, log.reaction)) +
  geom_point()
```



`group_by` to Group Data for `mutate`

Adds z-score based on each Subject

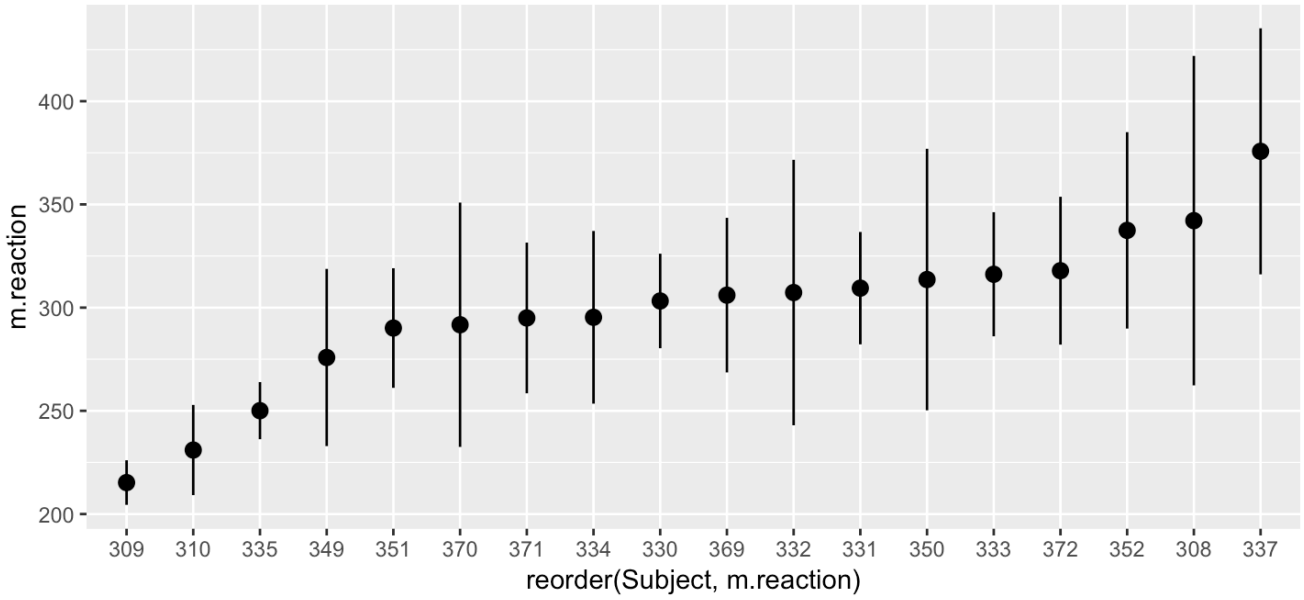
```
sleep.df = sleep.df %>% group_by(Subject) %>%
  mutate(z.reaction = scale(Reaction))

ggplot(sleep.df, aes(Days, z.reaction)) +
  geom_point()
```



`group_by` to Group Data for `summarise`

Aggregates data with summary statistics

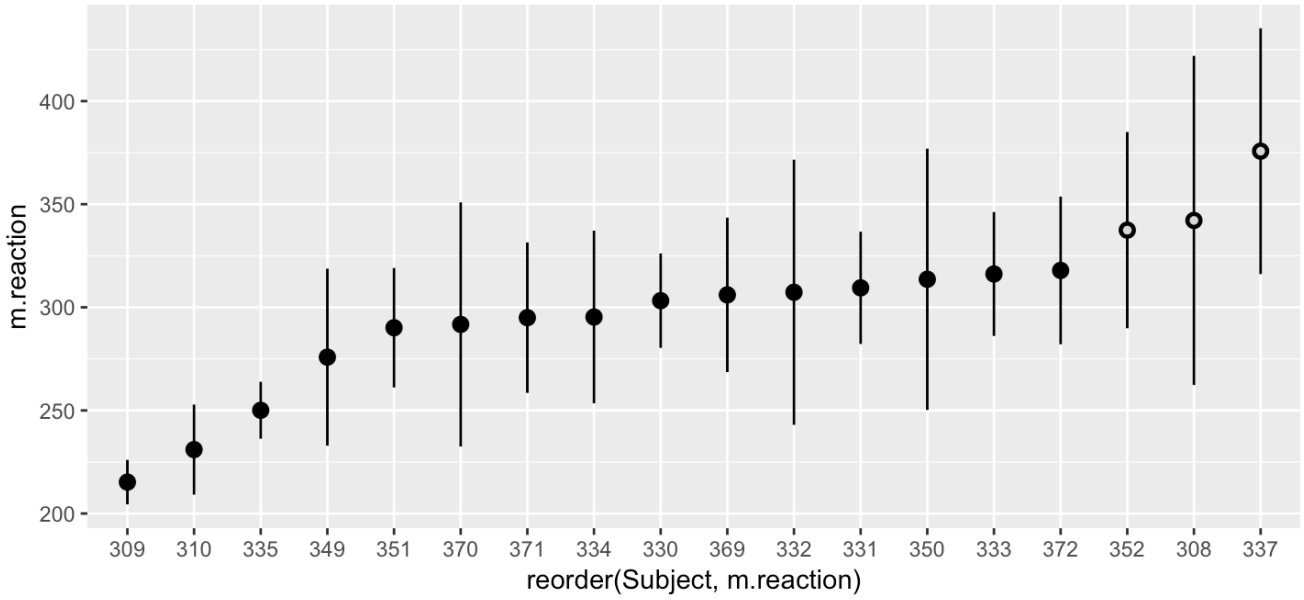
```
s.sleep.df = sleep.df %>% group_by(Subject) %>%
  summarise(m.reaction = mean(Reaction), sd.reaction = sd(Reaction))

ggplot(s.sleep.df, aes(reorder(Subject, m.reaction), m.reaction)) +
  geom_pointrange(aes(ymin = m.reaction-sd.reaction, ymax = m.reaction+sd.reaction))
```



`filter` to Subset the Data

Add layer to highlight outliers

```
r.s.sleep.df = s.sleep.df %>% filter(m.reaction>325)

ggplot(s.sleep.df, aes(reorder(Subject, m.reaction), m.reaction)) +
  geom_pointrange(aes(ymin = m.reaction-sd.reaction, ymax = m.reaction+sd.reaction)) +
  geom_point(data = r.s.sleep.df, size = 1, colour = "grey85")
```



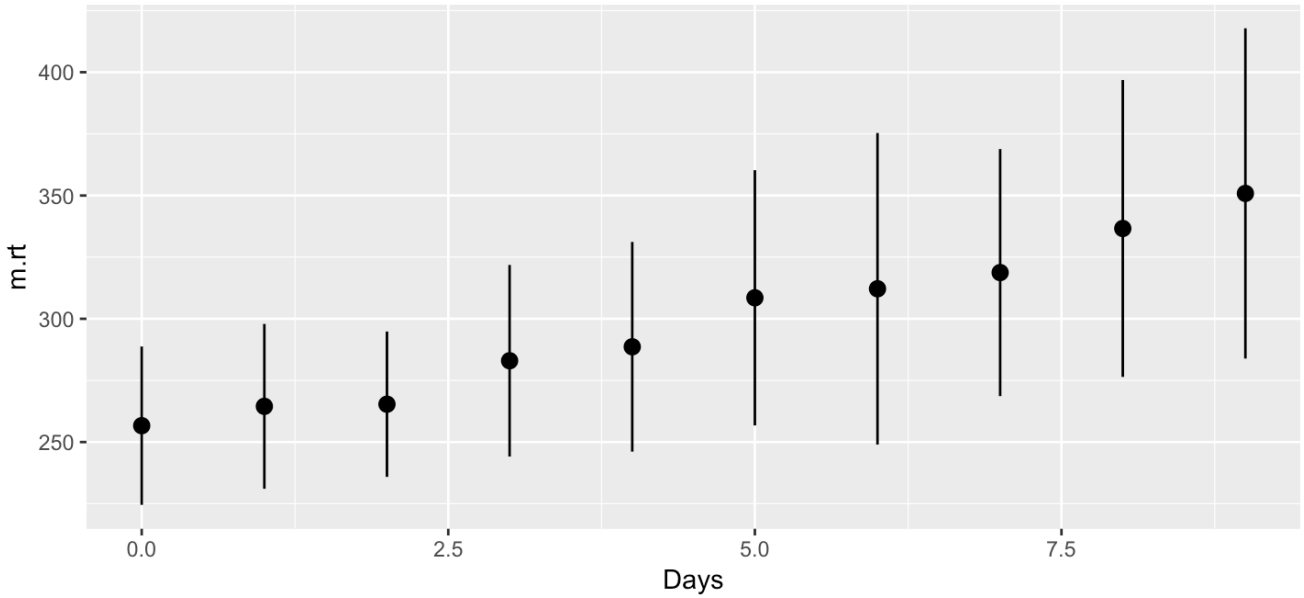
Exercise: Simple data aggregation

Calculate the mean and standard deviation of Reaction by day

– Group the data by `Days` and then summarise using `mean` and `sd`

– Plot the summary data using `geom_pointrange`

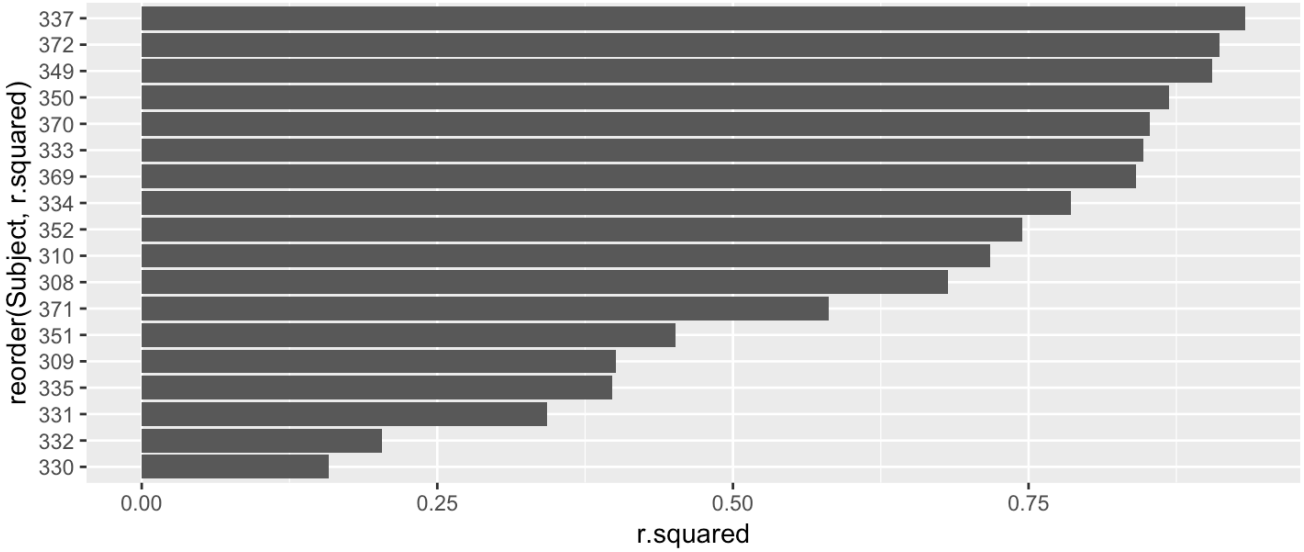
– Plot with 25th and 75th quantiles if this is too easy



do to Fit Model to Groups of Data

Fit regression model to each participant

```
models = sleep.df %>% group_by(Subject) %>%
  do(fit = lm(Reaction ~ Days, data = ., na.action = na.exclude))

s.model = models %>% glance(fit) # Extracts summary of models

ggplot(s.model, aes(reorder(Subject, r.squared), r.squared)) + geom_col() + coord_flip()
```



`join` to Combine Datasets

Mutating joins add columns from x and y

– `inner_join(x, y)` Returns all rows of x where there are matching values of y, all columns of x and y

– `left_join(x, y)` Returns all rows of x, all columns of x and y, rows of x with no match in y get NA

– `full_join(x, y)` Returns all rows and all columns of x and y, unmatched receive NA

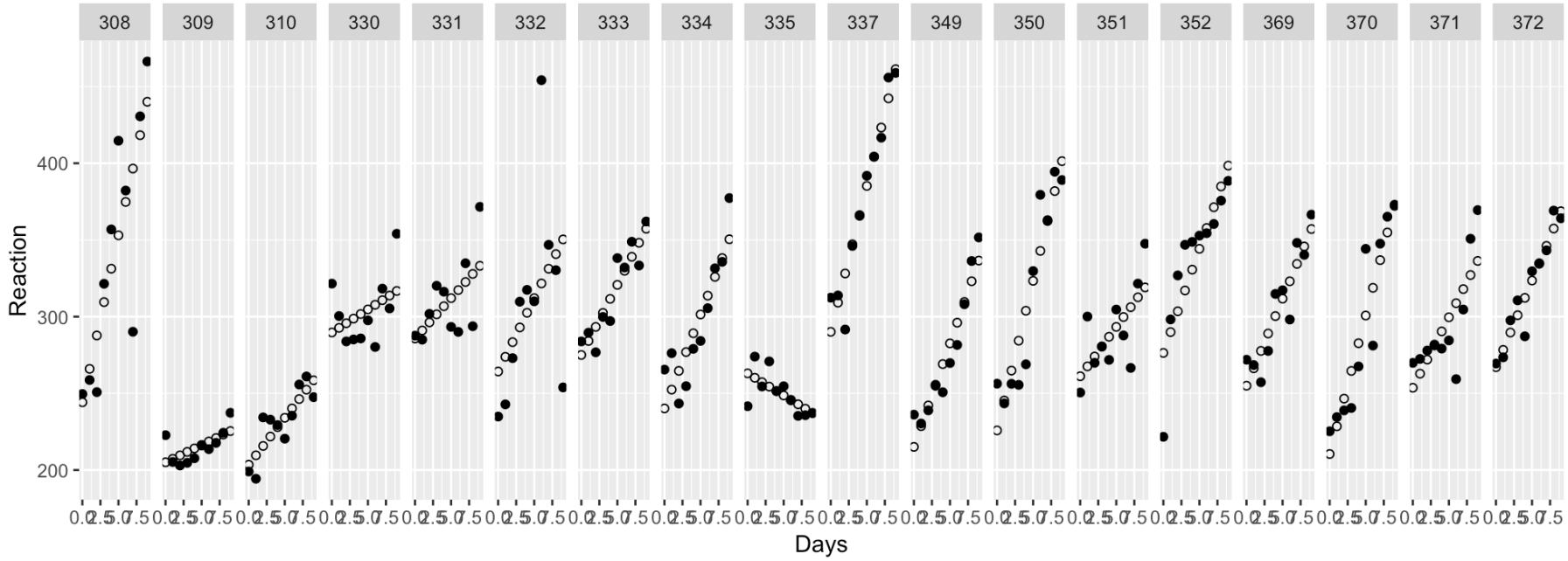
Filtering joins keep only columns from x

– `semi_join(x, y)` Return all row in x that have a matching values in y, columns of x

– `anti_join(x, y)` Return rows in x that are do NOT have a match in y, columns of x

`join` to Combine Datasets

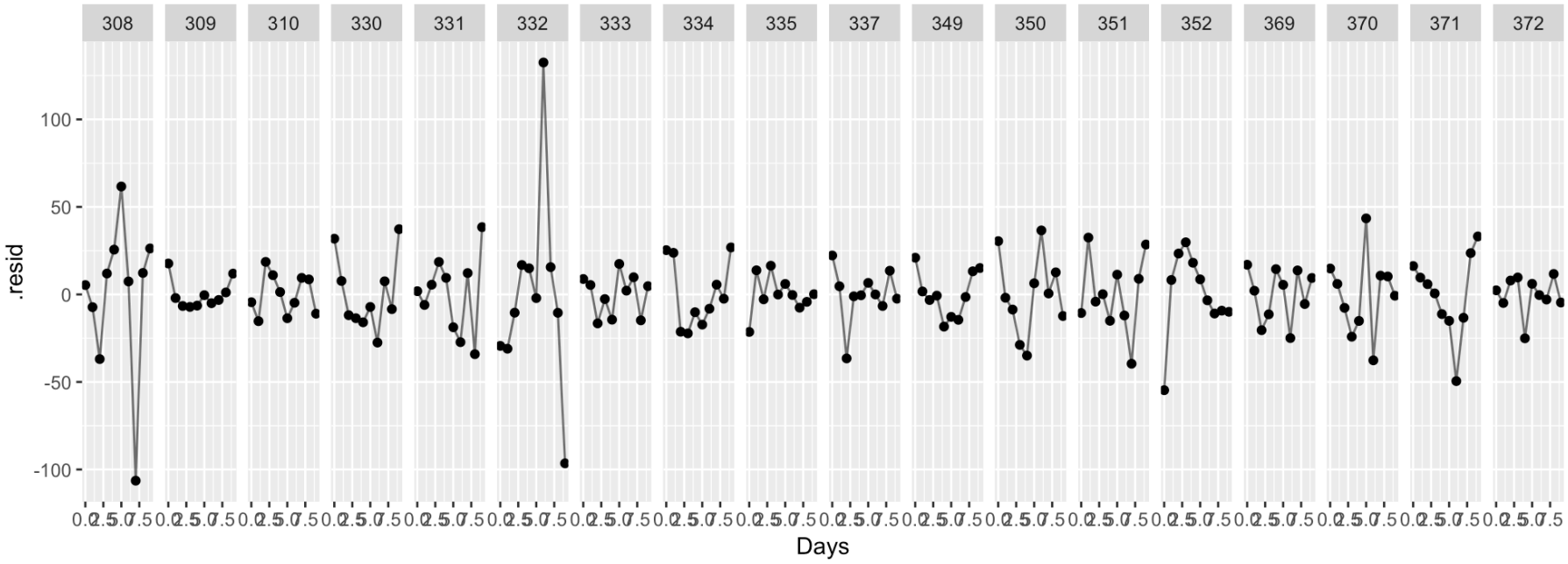
```
a.model.df = models %>% augment(fit) # Extracts summary of models
augmented.sleep.df = left_join(sleep.df, a.model.df, by = c("Subject", "Days", "Reaction"))

ggplot(augmented.sleep.df, aes(Days, Reaction)) +
  geom_point()+
  geom_point(aes(Days, .fitted), shape = 21) +
  facet_grid(.~Subject)
```



Pattern of Model Errors

Residuals highlight problems

```
ggplot(augmented.sleep.df, aes(Days, .resid)) +
  geom_point() +
  geom_line(alpha =.6) +
  facet_grid(.~Subject)
```



15/17

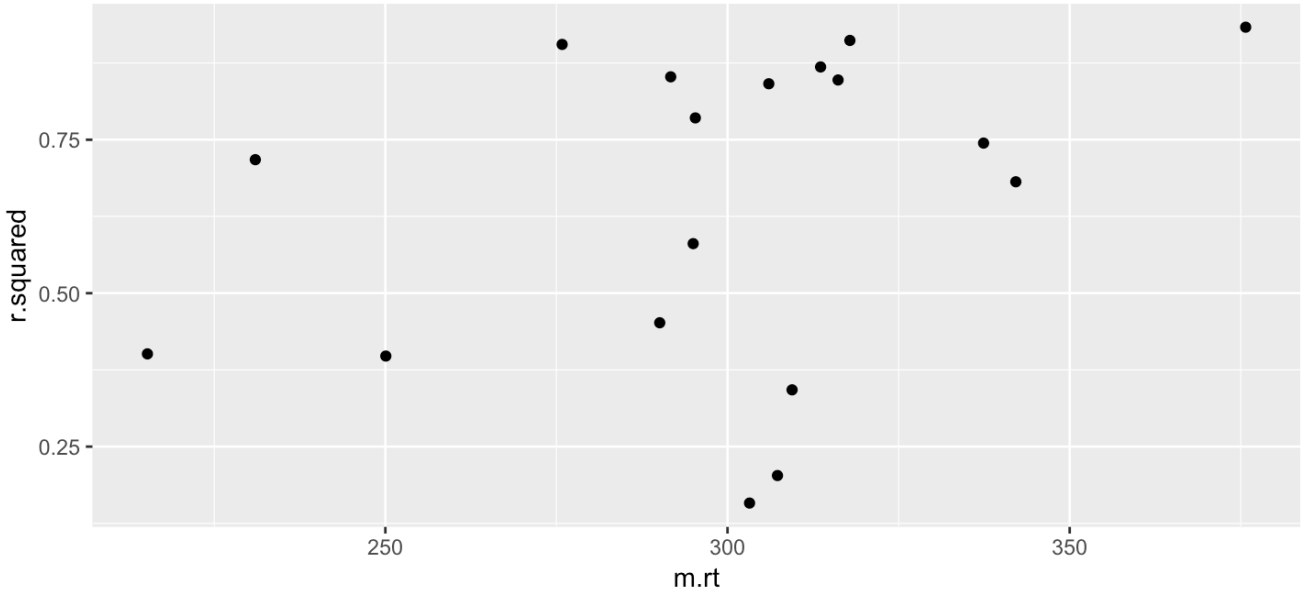
Exercise

Model and merge

– Fit a regression model for each participant and extract the r.squared value with the `glance` function

– Summarize the sleep data by person to calculate the mean reaction time

– Merge the summarized data and the r.square value and plot



dplyr for Data Reduction

dplyr as grammar of data reduction

- `select` to remove variables (i.e., columns)
- `filter` to subset the data and remove observations (i.e., rows)
- `mutate` to create new variables
- `summarise` to aggregate data across rows (i.e., mean of values)
- `group_by` to group variables for "mutate" and "summarise"
- `do` to fit a model or create a plot each group of the data
- `join` to combine datasets

Pipe (i.e., `%>%`) for combining operations in a comprehensible way