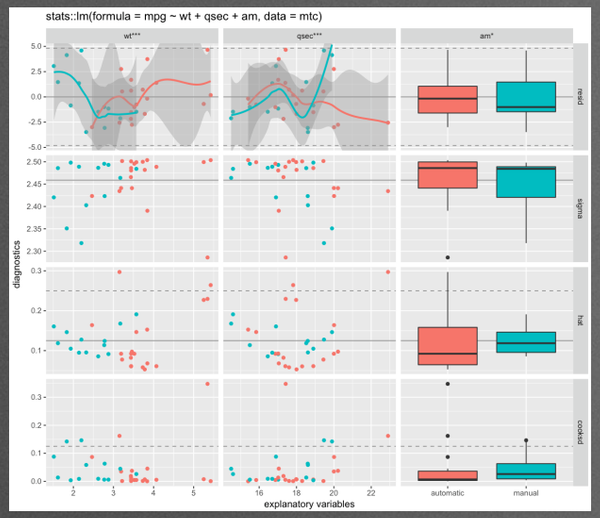
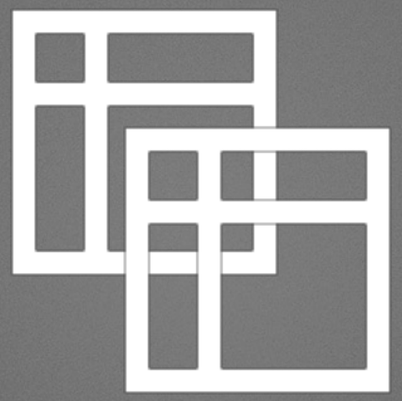
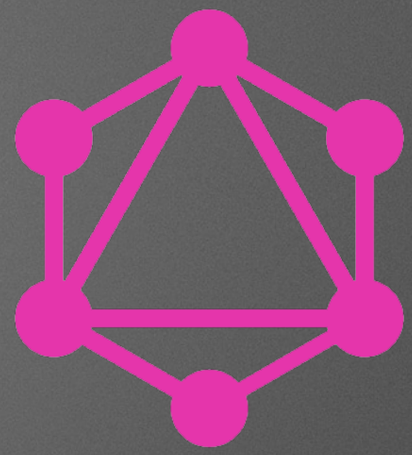






Generalized Plot Matrices, Automatic Cognostics, and Efficient Data Exploration

Barret Schloerke
Statistics PhD Candidate
Purdue University

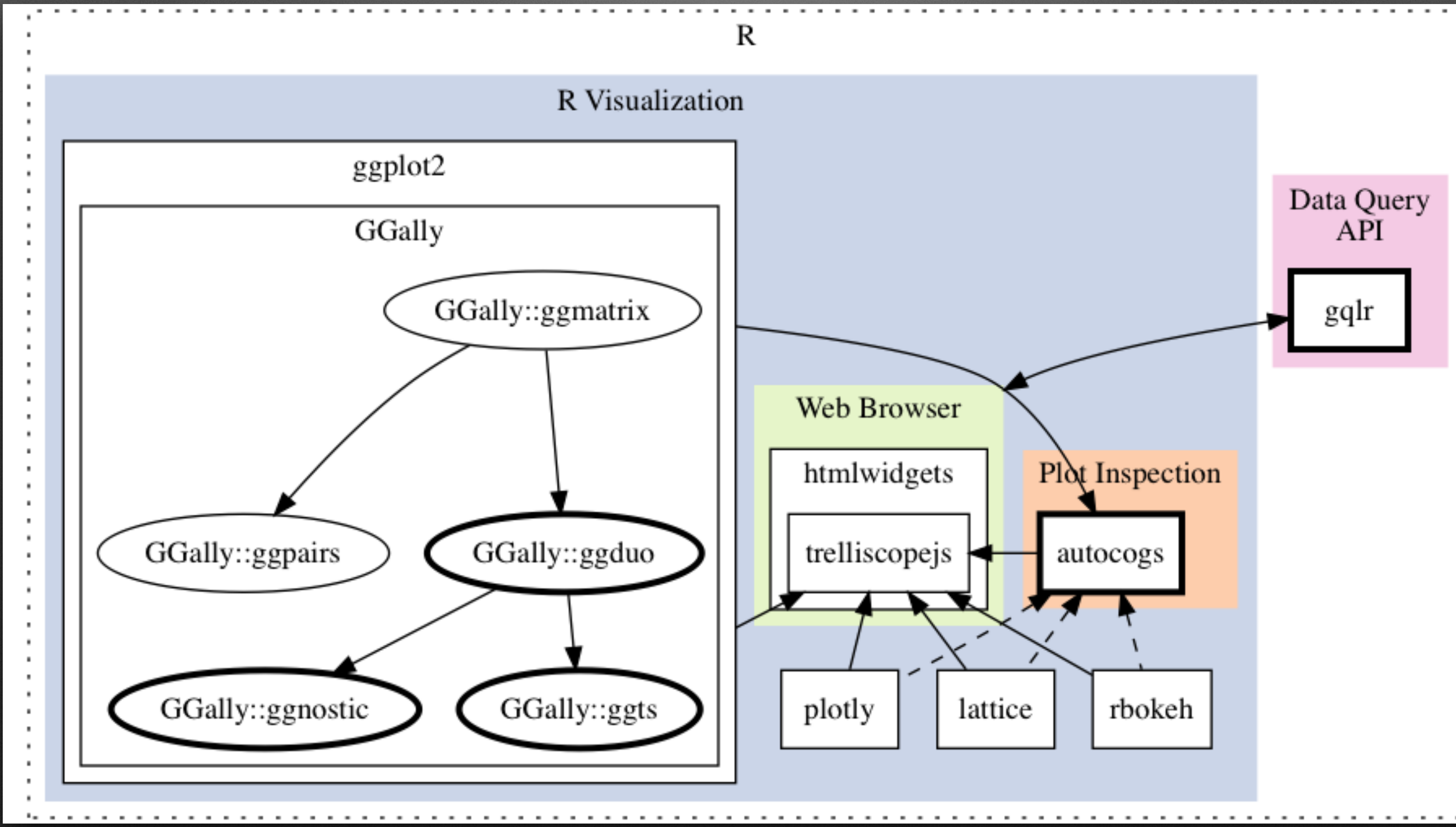
Dr. William Cleveland (Co-Chair)
Dr. Ryan Hafen (Co-Chair)
Dr. Bowei Xi
Dr. Vinayak Rao

About Me

- **Purdue University**
  - 5th Year Statistics PhD Candidate
  - Chairs: Dr. William Cleveland and Dr. Ryan Hafen
  - Research in large data visualization using R
- Metamarkets.com - 1.5 years
  - Front end engineer
- **Iowa State University**
  - B.S. in Computer Engineering
  - Research in statistical data visualization with R

Generalized Plot Matrices, Automatic Cognostics, and Efficient Data Exploration

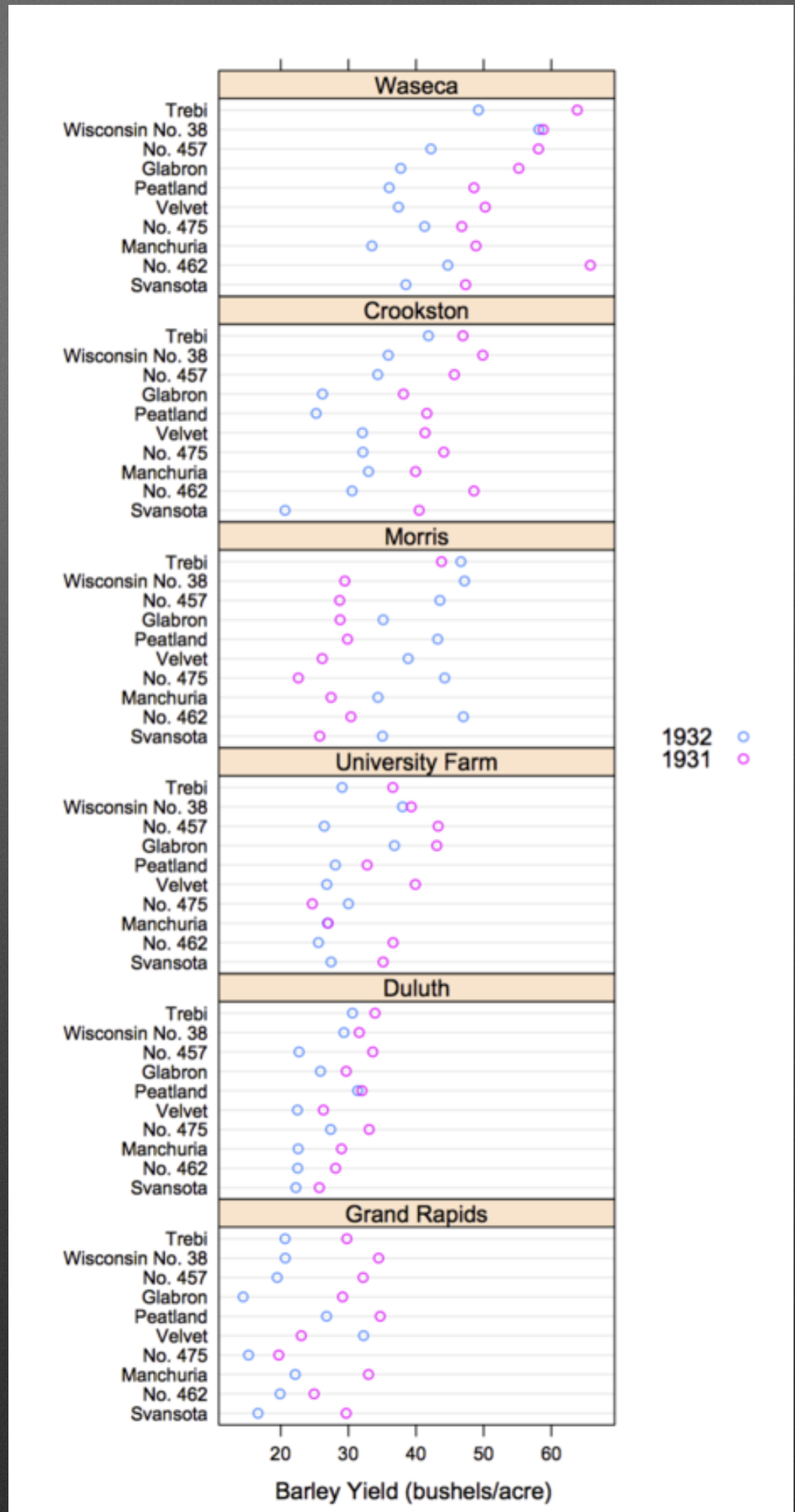
Topic Organization



Generalized Plot Matrices

Trellis Display

- Data is split into meaningful subsets, usually conditioning on variables of the dataset
- A visualization method is applied to each subset
- The image for each subset is called a "panel"
- Panels are arranged in an array of rows, columns, and pages, resembling a “garden trellis”
- `facet_*()`'ing in `ggplot2`



Why Trellis is Effective

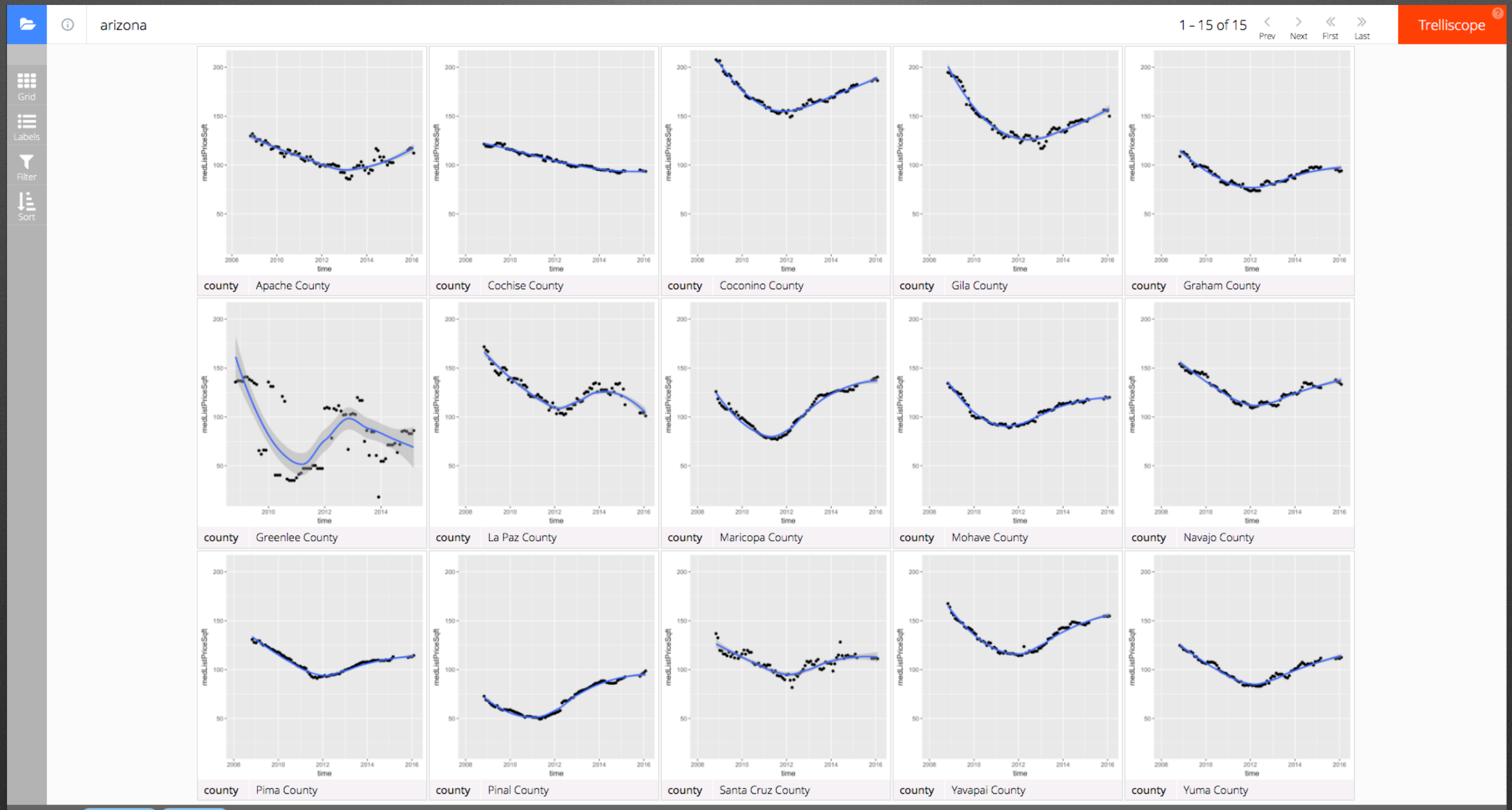
- Flexible to create
  - Data complexity / dimensionality / size
    can be handled by splitting the data into subsets
  - Complete freedom with what is plotted in every panel
- Effective to consume
  - Understand one panel —> Understand every panel
  - Scanning across panels elicits comparisons to reveal repetition and change, pattern and surprise

Example / Data Description

- Monthly median home listing and number of units sold for 2,984 counties in the contiguous United States From 2008 to January 2016,
- Harvested from Quandl's Zillow

```
> housing %>% dplyr::group_by(county, state)
Source: local data frame [247,082 x 7]
Groups: county, state [2,984]

    fips              county  state       time  nSold medListPriceSqft
*  <fctr>              <fctr> <fctr>     <date>  <dbl>             <dbl>
1   06037 Los Angeles County     CA 2008-01-31 505900                NA
2   06037 Los Angeles County     CA 2008-02-29 497100                NA
3   06037 Los Angeles County     CA 2008-03-31 487300                NA
4   06037 Los Angeles County     CA 2008-04-30 476400                NA
5   06037 Los Angeles County     CA 2008-05-31 465900                NA
6   06037 Los Angeles County     CA 2008-06-30 456000                NA
7   06037 Los Angeles County     CA 2008-07-31 445700                NA
8   06037 Los Angeles County     CA 2008-08-31 435300                NA
9   06037 Los Angeles County     CA 2008-09-30 426700                NA
10  06037 Los Angeles County     CA 2008-10-31 419800          273.3073
# ... with 247,072 more rows, and 1 more variables: medSoldPriceSqft <dbl>
```

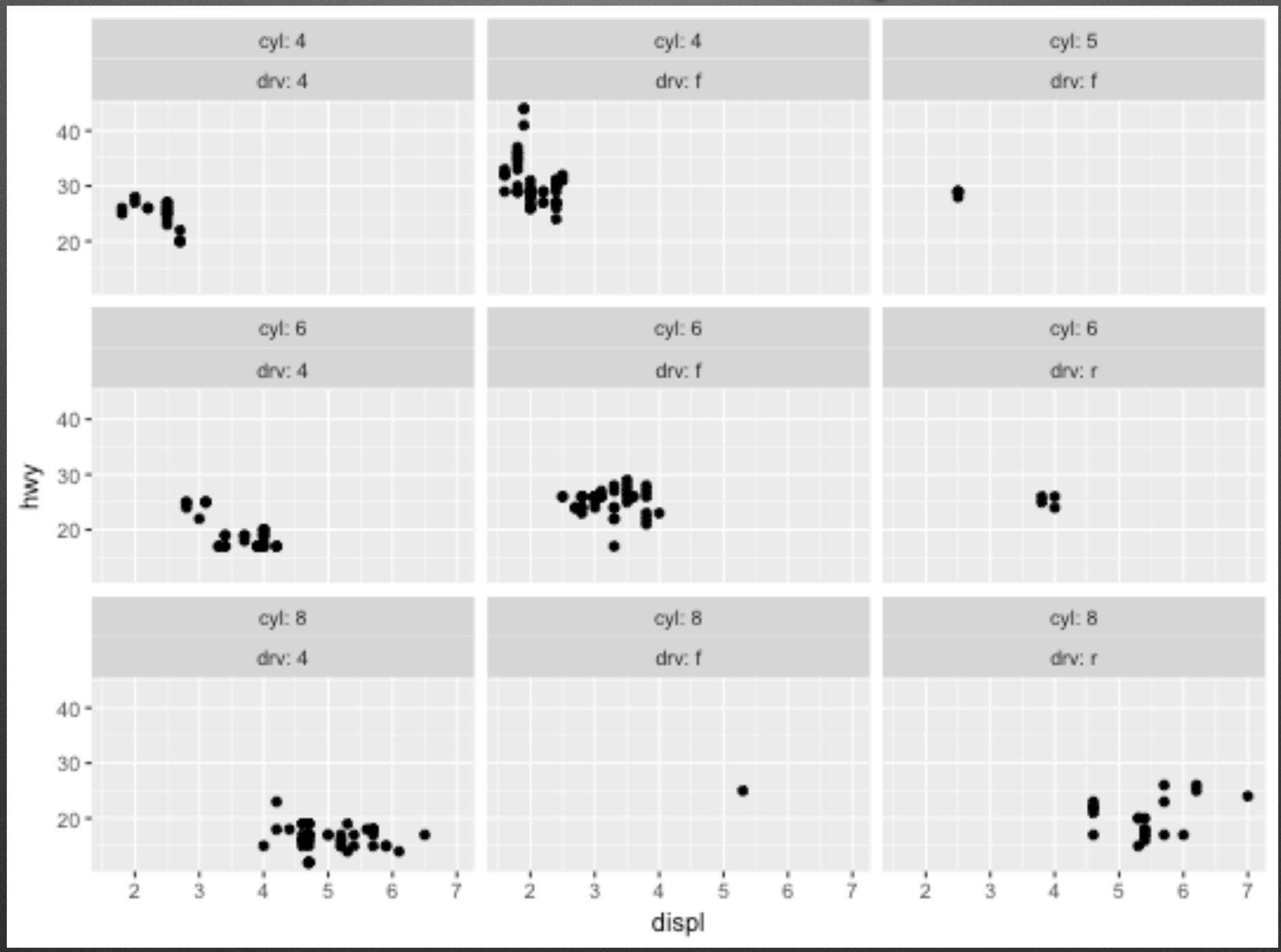
Housing: Arizona



ggplot2

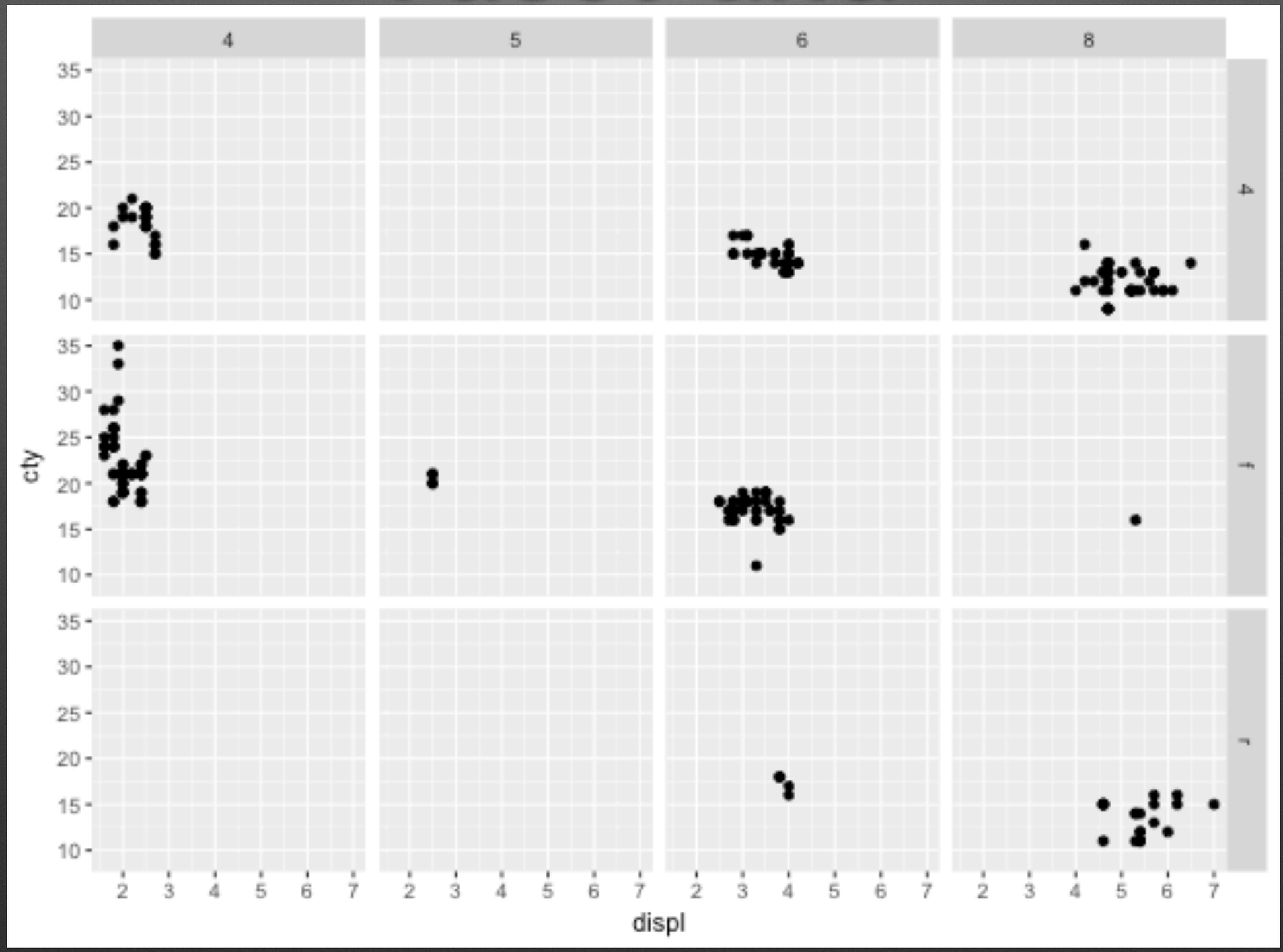
- Layered Grammar of Graphics
- Implements Trellis Display
  - Not as robust as `lattice`
    - `facet_wrap()`
      - No required structure
    - `facet_grid()`
      - Layout is determined by levels of variables used

Facet Wrap



http://ggplot2.tidyverse.org/reference/facet_wrap.html

Facet Grid



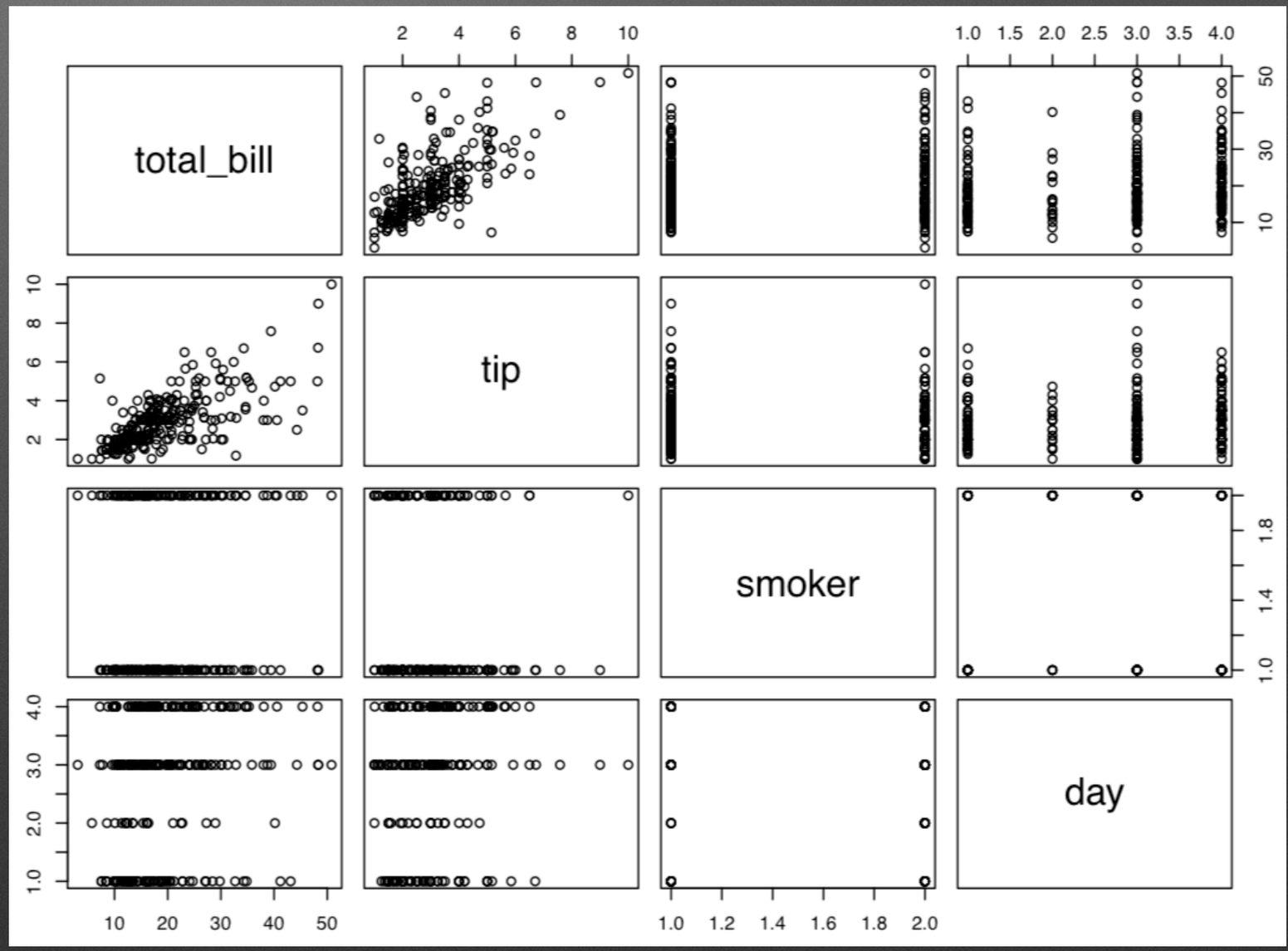
http://ggplot2.tidyverse.org/reference/facet_grid.html

Trellis Display Rules

- Each panel contains conditioned data
  - (typically) independent
- Each panel contains the same plotting layers
- Each panel displays the same X and Y data columns

Pairs Plot

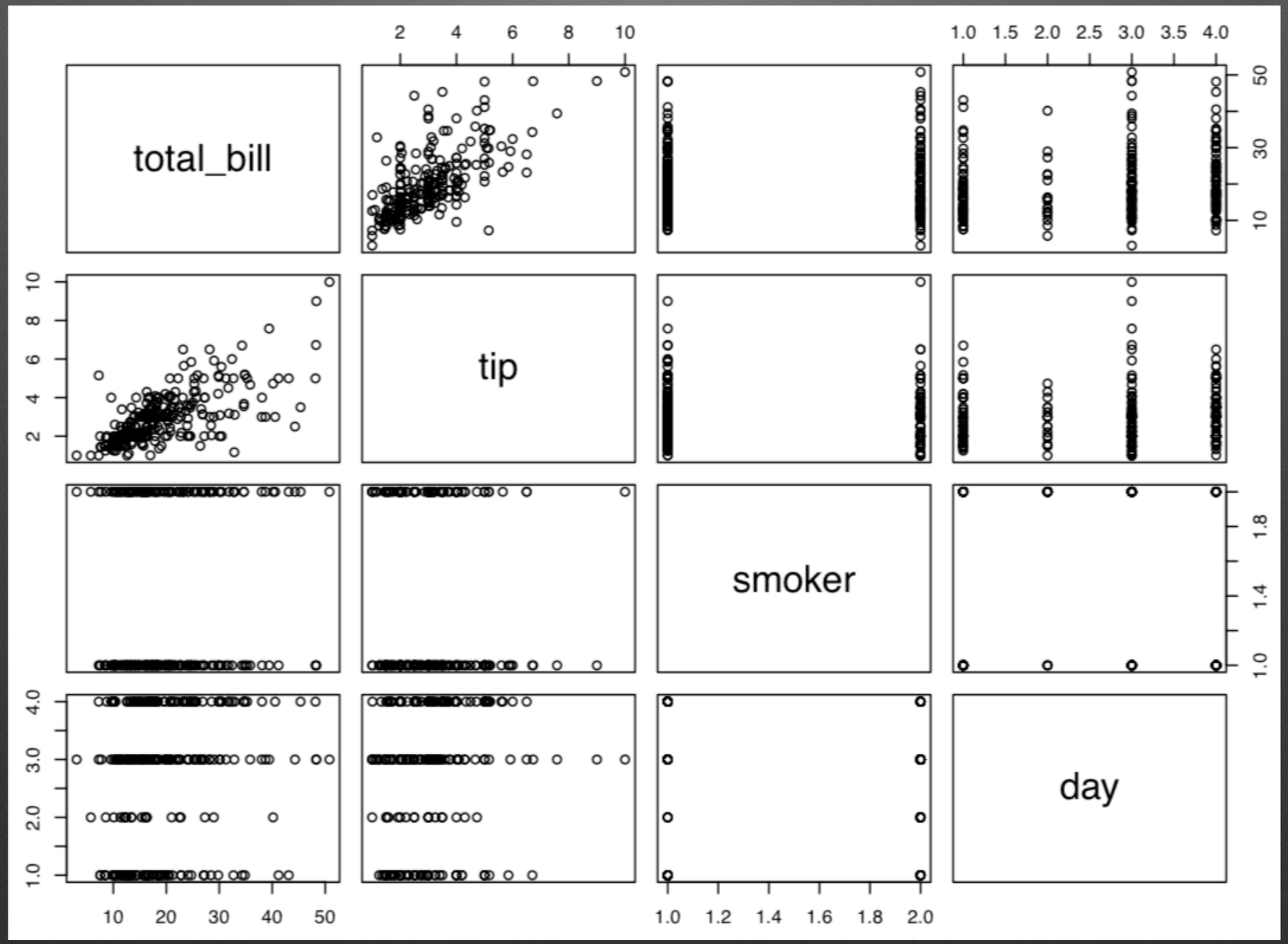
- No conditioning
- All X and Y variable combinations
- Between plot relationships



Traditional Pairs Plot Rules

- Each panel contains the same rows of data
- Each panel contains the same plotting layers
- Each panel displays all X and Y data column combinations

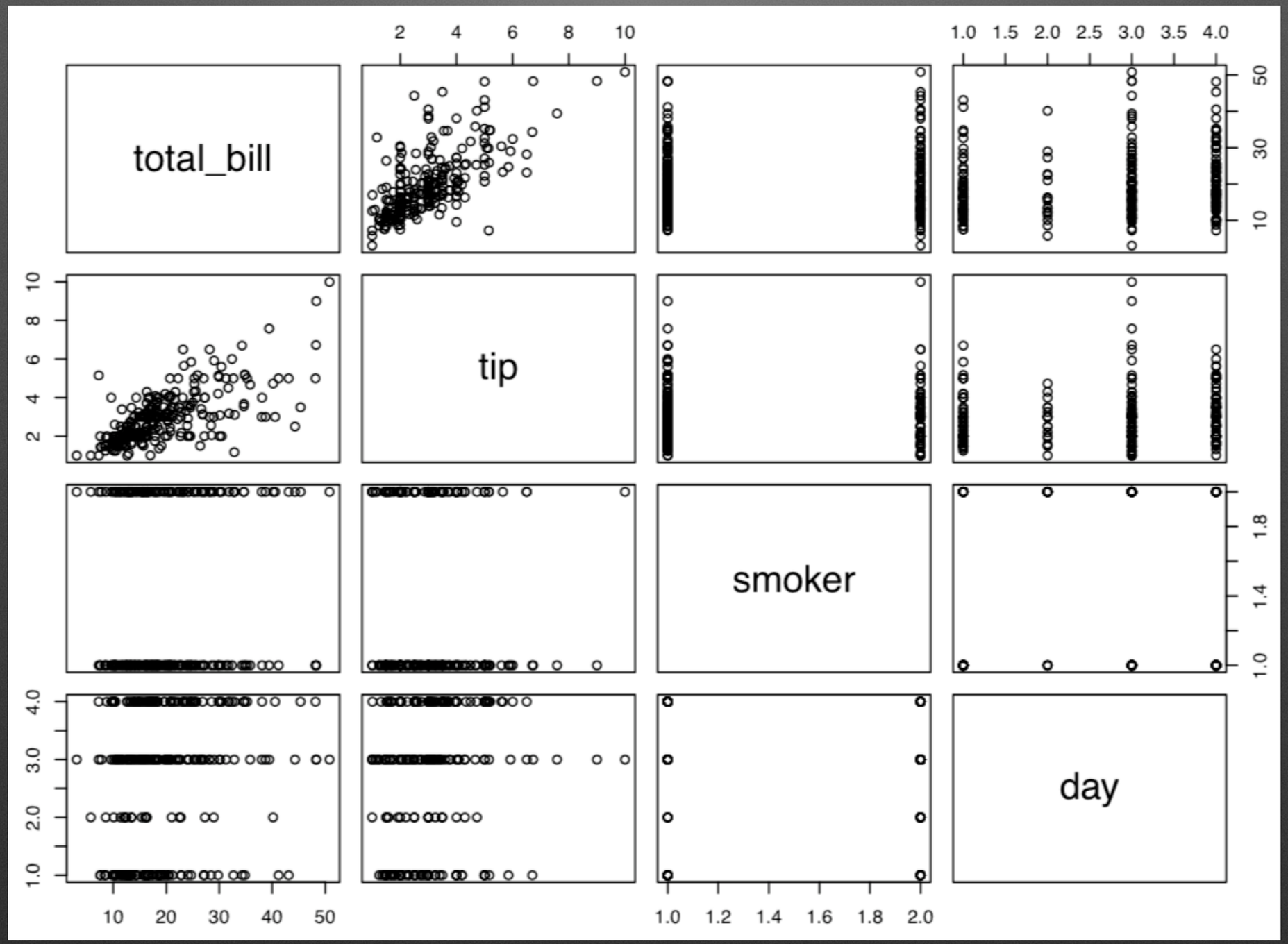
Traditional Pairs Plot



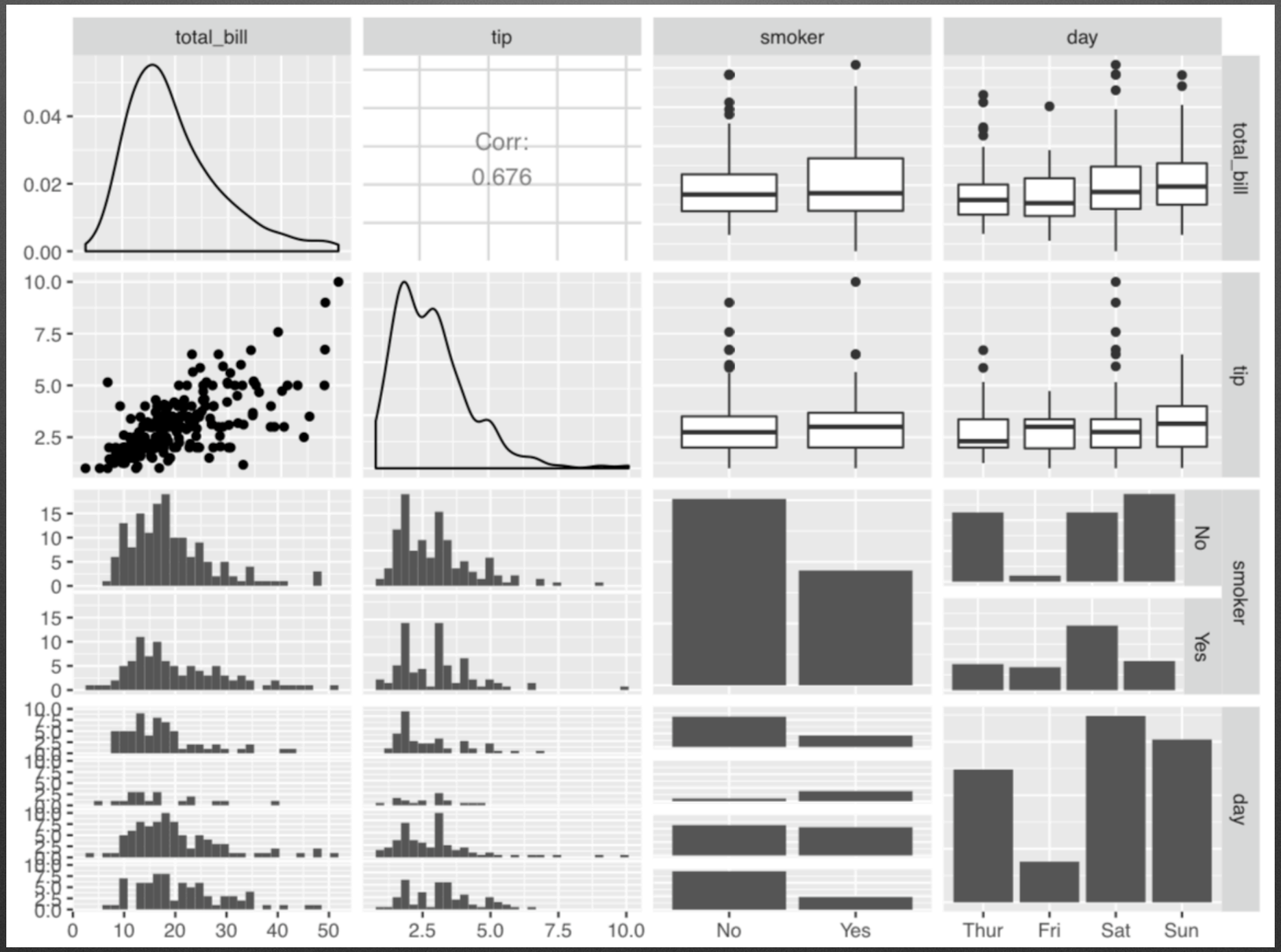
Generalized Pairs Plot Rules

- Each panel contains same rows of data
- Each panel contains appropriate plotting layers
- Each panel displays all X and Y data column combinations

Traditional Pairs Plot



Generalized Pairs Plot



GGally::ggpairs

- Emerson, Green, Schloerke, Crowley, Cook, Hofmann, Wickham. "The Generalized Pairs Plot." *JCGS*, vol. 22, no. 1, pp. 79-91, 2012.
- Complete pairwise plot matrix
  - A, B, C vs. A, B, C
- Three "matrix" sections:
  - `upper`, `lower`, `diag`
- Three main section types:
  - continuous, combo, discrete
- Produces a `ggmatrix` object


```
pm <- ggpairs(
  tips, c(1,2,6),
  mapping = aes(color = sex)
); pm
```

`GGally::ggmatrix` structure

- Generic plot matrix with fine tune control of
  - Axis labels in plot strips
  - Overall X and Y axis titles and plot matrix title
  - May have a variable number of rows (n) and columns (m)
  - Contains a list made of
    - (custom) `ggplot2` objects
    - Functions that will evaluate with the supplied data
  - Allows for many plots (`n*m`) with larger data

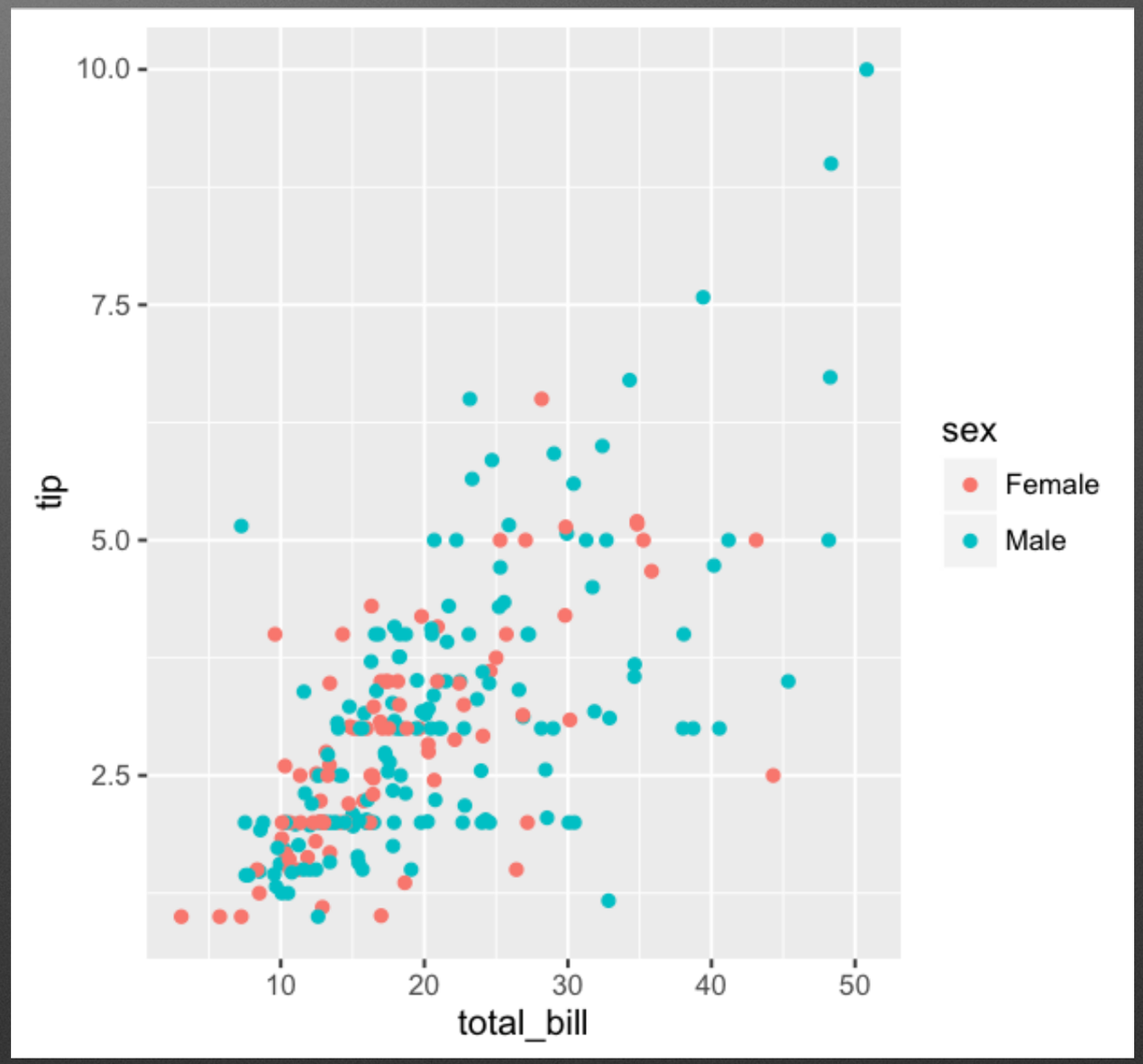
GGally::ggmatrix

- Retrieve individual plot

ggplot2_obj <- pm[3,1]

- Store individual plot

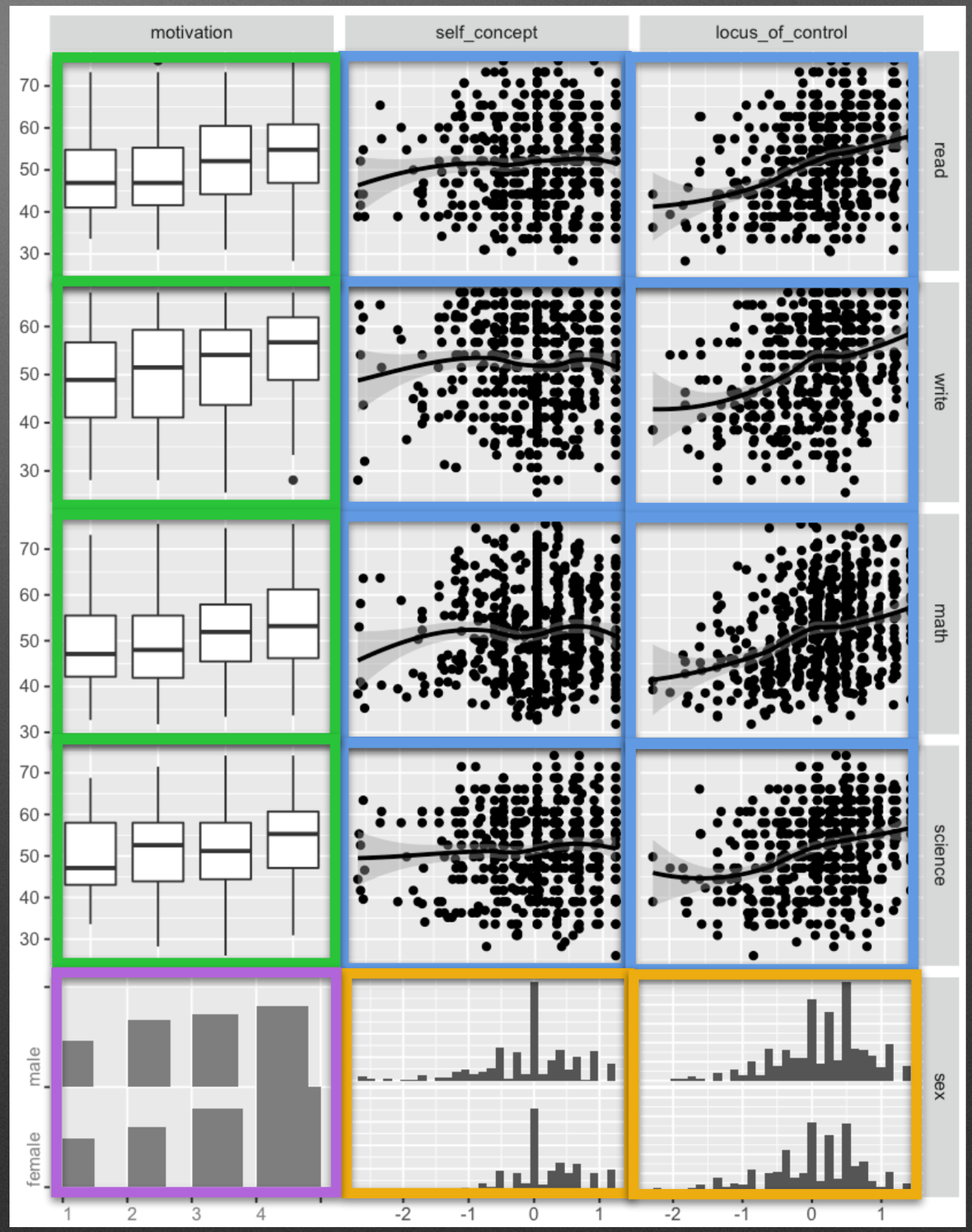
pm[row_pos, col_pos] <-
other_ggplot2_obj



pm[3,1]

GGally::ggduo

- Pairwise plot matrix for two-grouped data
  - A, B, C vs. D, E, F, G, H
- Four main types:
  - continuous
  - comboVertical
  - comboHorizontal
  - discrete




```
ggduo(psychademic, 3:1, 4:8, showStrips = FALSE)
```

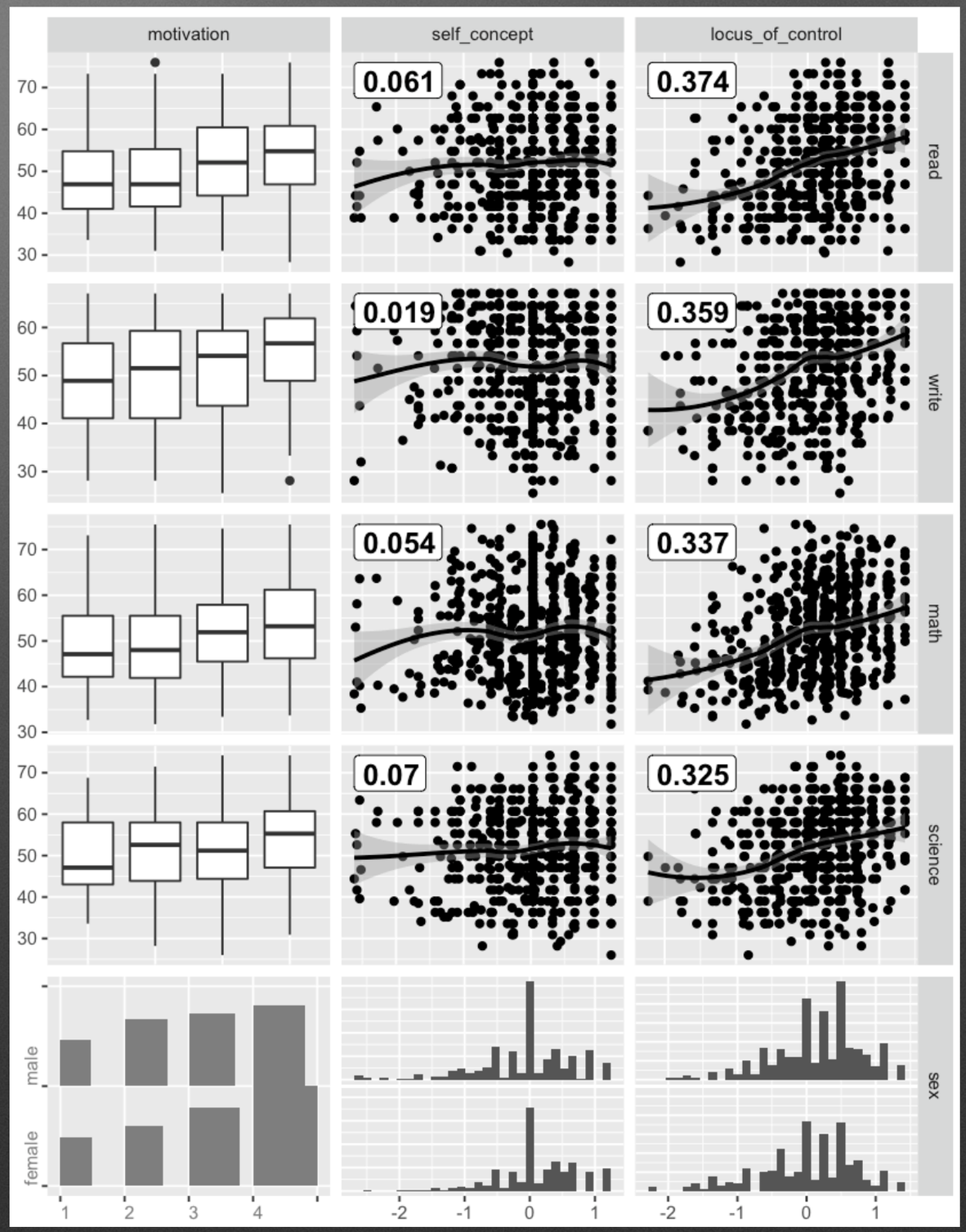
Application

- Can directly be used in
  - Canonical correlation analysis
  - Multiple time series analysis
  - Regression diagnostics

Canonical correlation analysis (CCA)

- `GGally::ggpairs` can be used for “within” correlation
- `GGally::ggduo` is useful for “between” correlations

```
ggduo(
  psychademic, 1:3, 4:8,
  types = list(
    continuous = my_loess_with_cor
  ),
  showStrips = FALSE
)
```

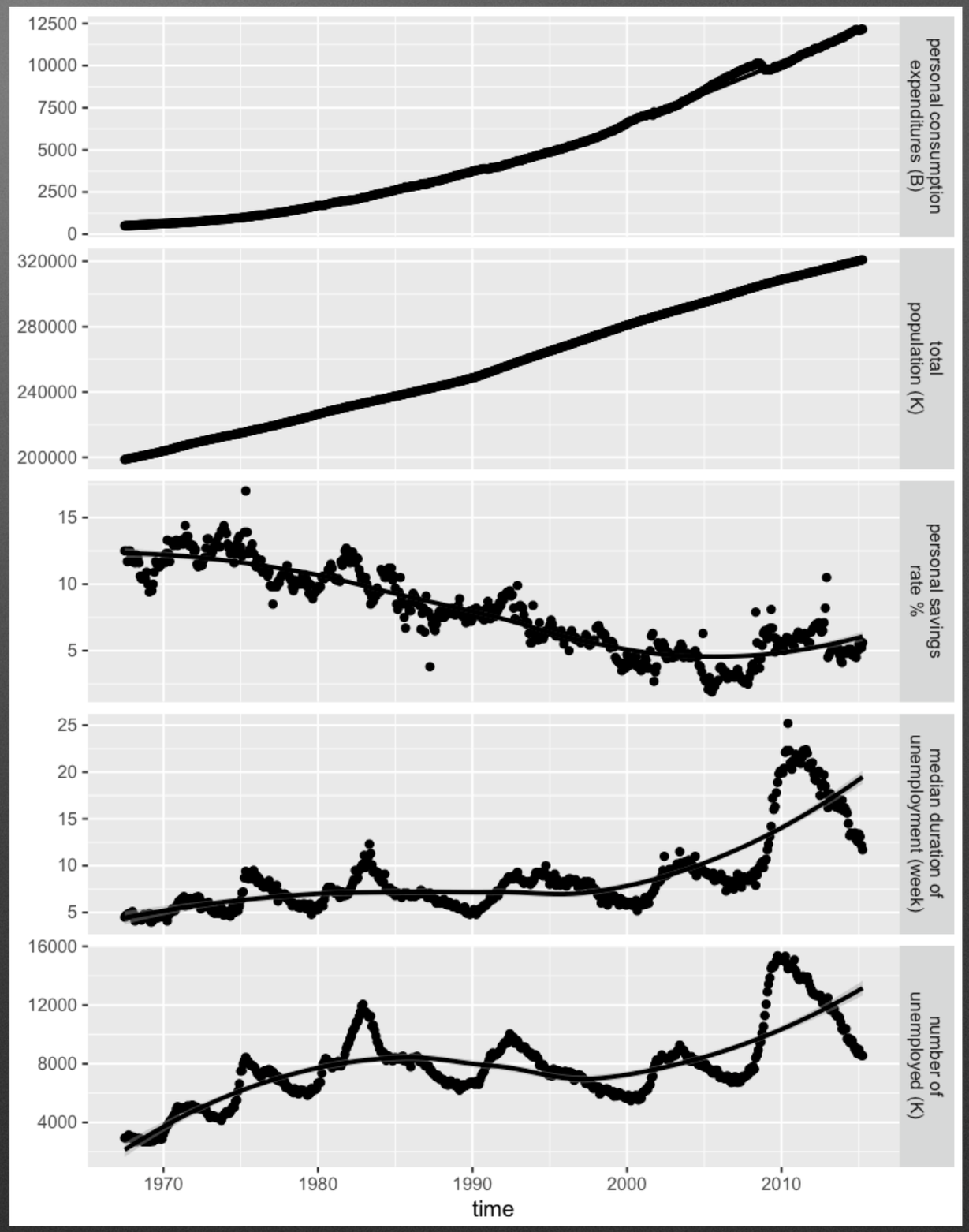


`ggduo(psychademic, 3:1, 4:8, showStrips = FALSE)`

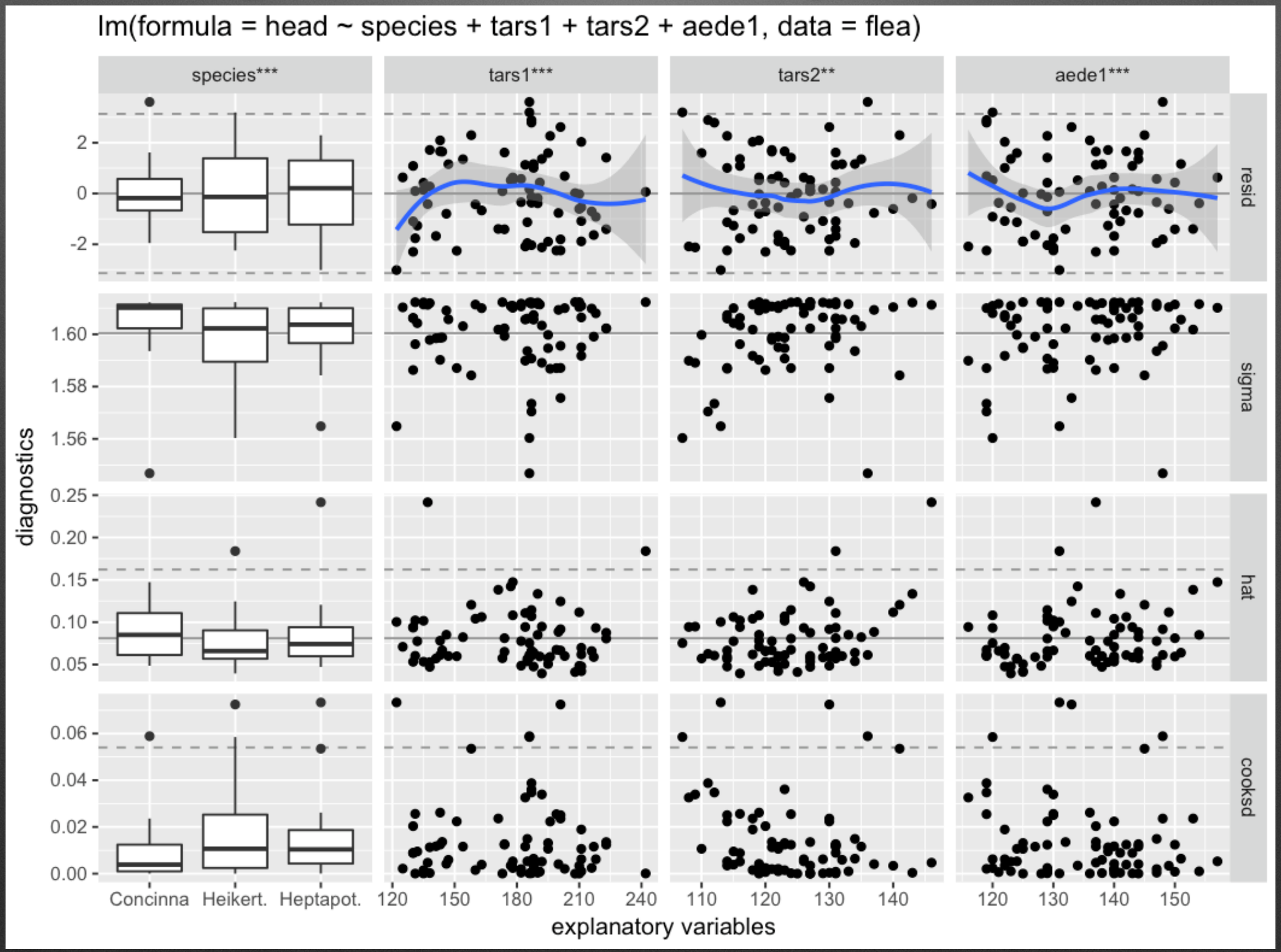
Multiple time series

- Multiple Y variables over time

```
ggts(
  economics, 1, 2:6,
  columnLabelsY = y_labels
)
```



Regression Diagnostics



Regression Diagnostics

- Model diagnostics already exist (independently)
- Common (data length) model diagnostics
  - Residuals
  - Leave-one-out sigma value
  - Influence (Hat matrix diagonal)
  - Cook's Distance

ggnostic

- Display all model diagnostics against each explanatory variable
- Provide **significance** reference lines when appropriate
  - Residuals: $1.96 * N(0, \sigma^2); E(Residuals_i) = 0$
  - Leverage: $2 * p/n; E(Leverage_i) = p/n$
  - Cooks's distance: $F_{p,n-p}(0.5)$
- Display Anova F test output for each explanatory variable

Linear Model Diagnostics Example

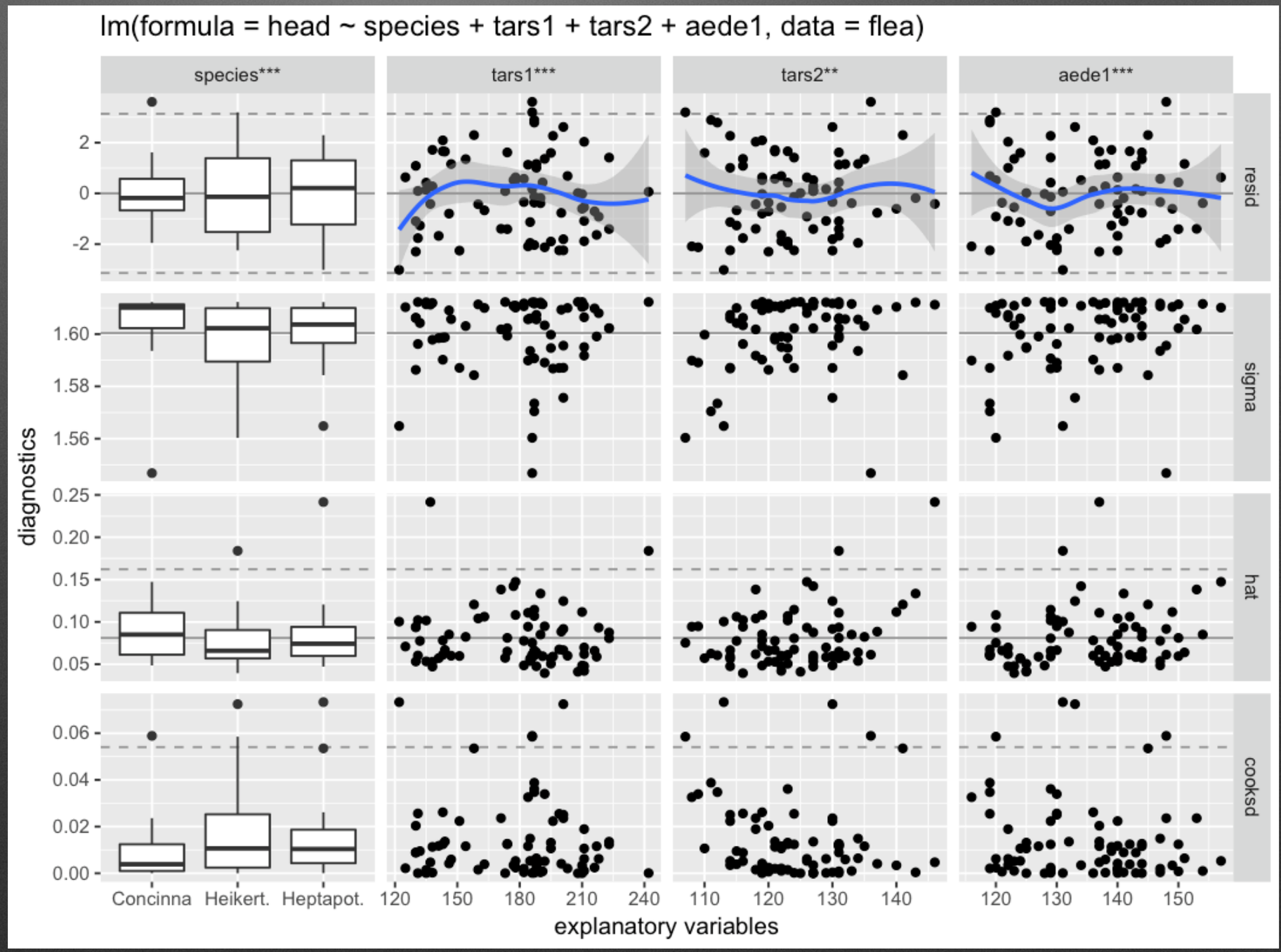
```
mod <- step(
  lm(
    formula = head ~ .,
    data = flea
  )
)
mod
# Call:
# lm(formula = head ~ species + tars1 + tars2 + aede1, data = flea)
#
# Coefficients:
#      (Intercept)   speciesHeikert.  speciesHeptapot.             tars1
#          0.63693           1.13651           5.22207           0.06815
#            tars2             aede1
#          0.10160           0.17069
```

Linear Model Diagnostics Examples

```
summary(mod)
# Call:
# lm(formula = head ~ species + tars1 + tars2 + aede1, data = flea)
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -3.0152 -1.2315 -0.0048  1.1490  3.6068
#
# Coefficients:
#                  Estimate Std. Error t value Pr(>|t|)
# (Intercept)       0.63693    6.09162   0.105 0.917034
# speciesHeikert.   1.13651    1.25452   0.906 0.368171
# speciesHeptapot.  5.22207    0.92110   5.669 3.18e-07 ***
# tars1             0.06815    0.01989   3.426 0.001043 **
# tars2             0.10160    0.03297   3.081 0.002974 **
# aede1             0.17069    0.04275   3.993 0.000163 ***
# —
# Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#
# Residual standard error: 1.6 on 68 degrees of freedom
# Multiple R-squared:  0.685,  Adjusted R-squared:  0.6618
# F-statistic: 29.57 on 5 and 68 DF,  p-value: 7.997e-16
```

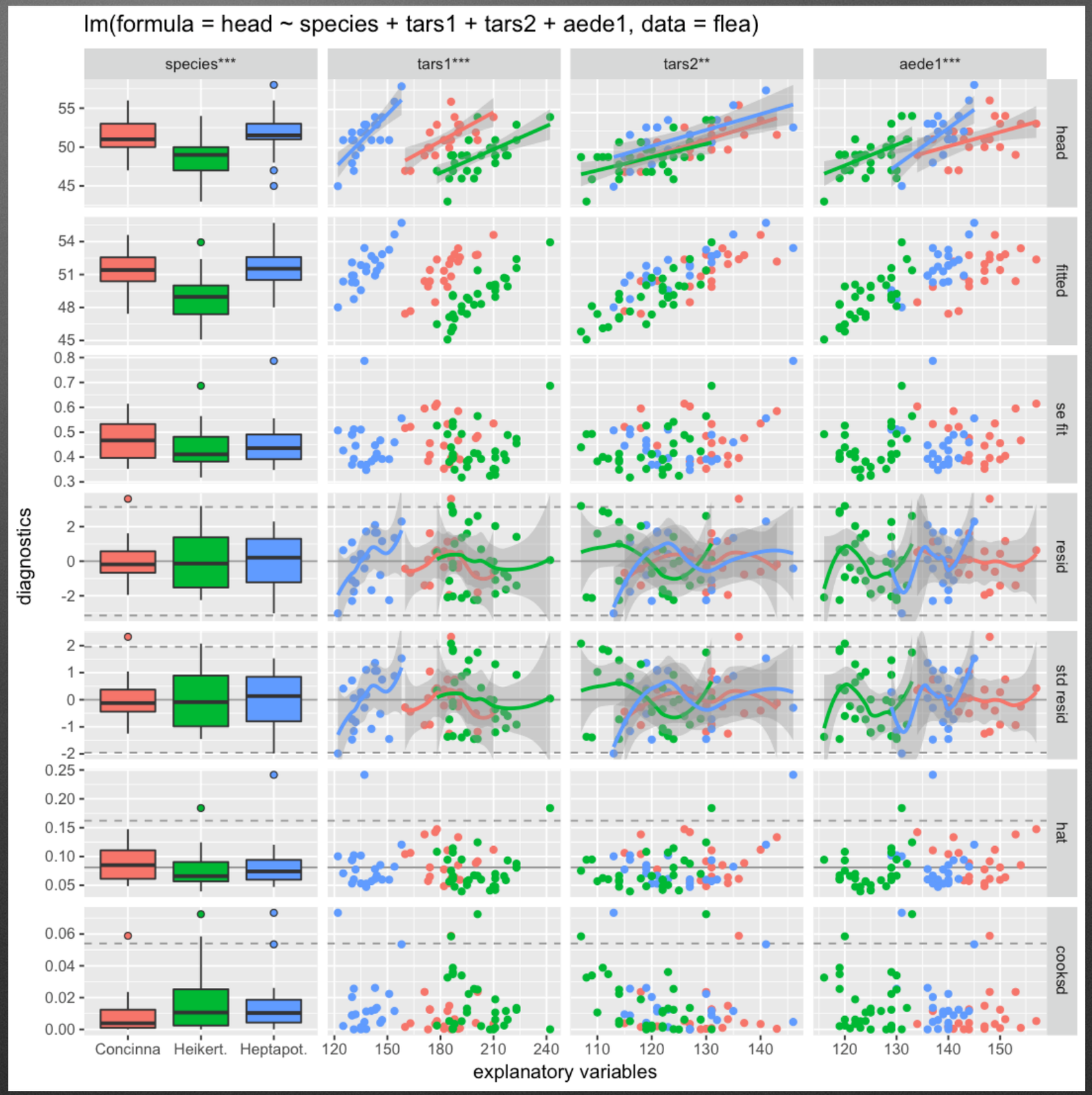
ggnostic

```
ggnostic(mod)
```



ggnostic

```
ggnostic(
  mod,
  aes(color = species),
  columnsY = c(
    "head",
    ".fitted",
    ".se.fit",
    ".resid",
    ".std.resid",
    ".hat",
    ".cooksd"
  )
)
```



Generalized Plot Matrix for Two-Grouped Data

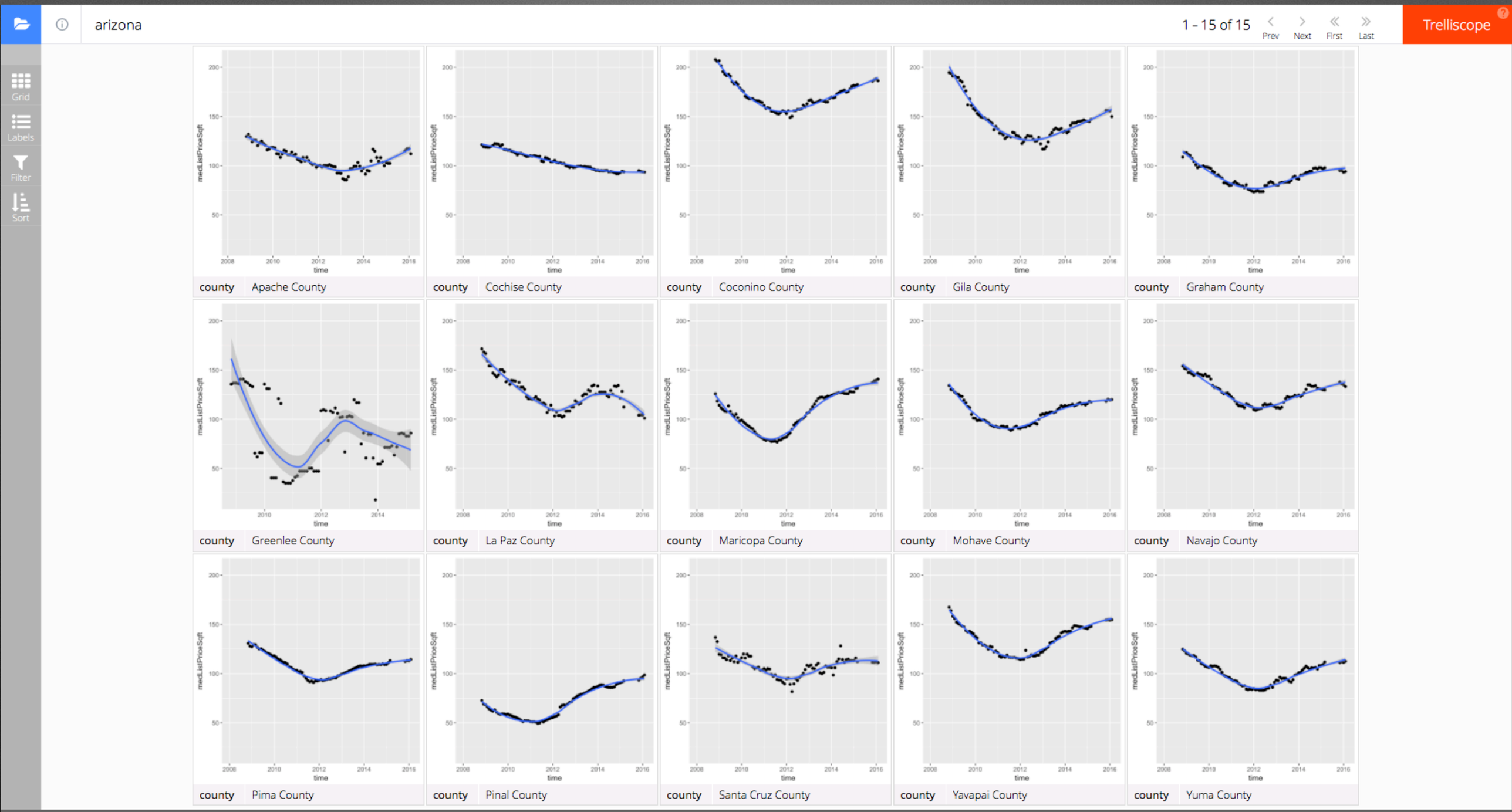
- Utilizes the Generalized Plot Matrix framework
  - Display the same rows of information
  - Display data with appropriate visualization method
  - Display all X and Y variable combinations

Plot Matrix Data

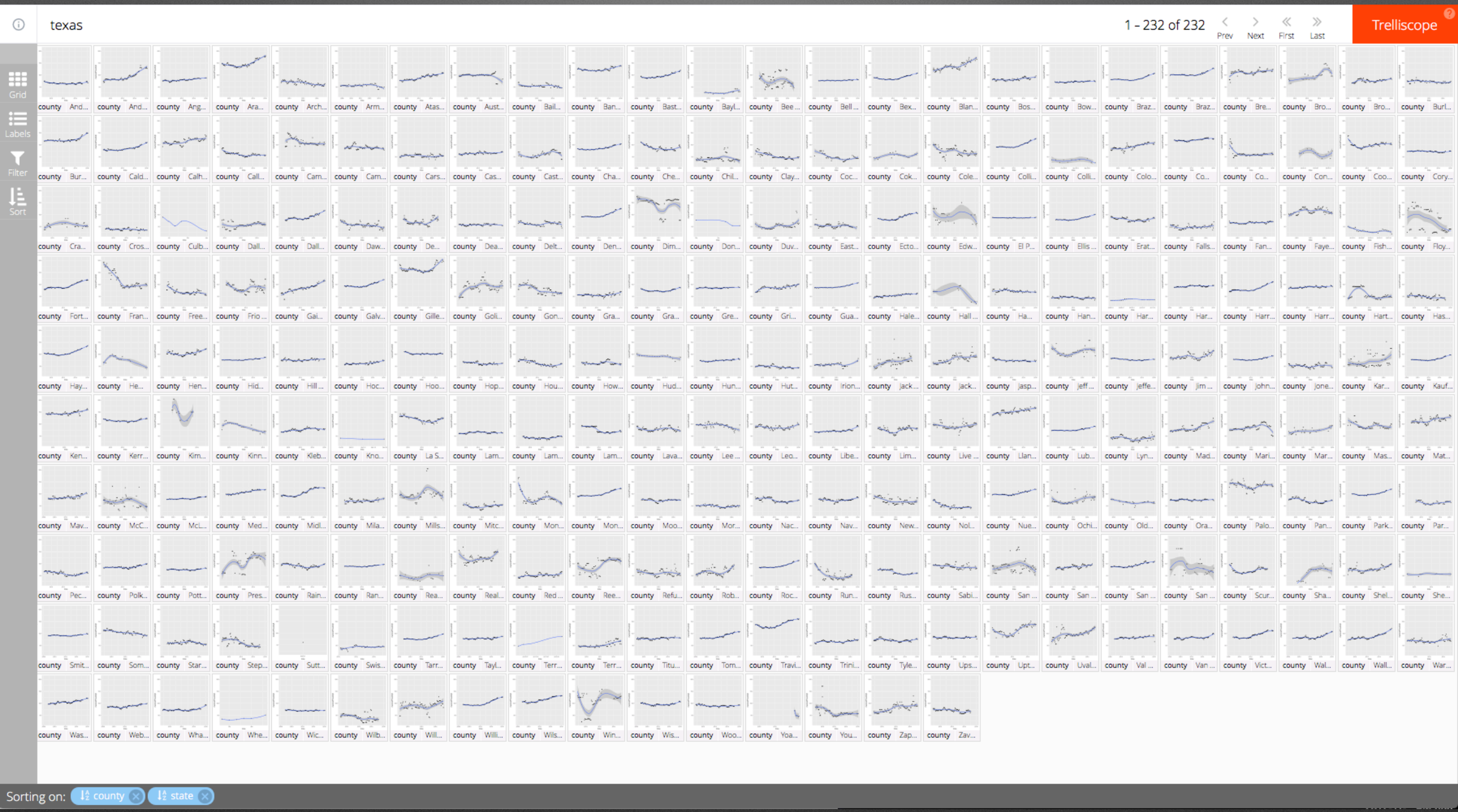
```
tips <- reshape::tips

economics <- ggplot2::economics

psychademic <- GGally::psychademic

flea <- GGally::flea
```

Automatic Cognostics

Trellis Display: Arizona Housing



Trellis Display: Texas Housing



Scaling Trellis

- Large data lends itself nicely to the idea of Trellis Display
  - Typically comprised of collections of smaller data from many subjects, locations, time periods, etc.
  - Natural to break the data up based on these dimensions and make a plot for each subset
- Potentially hundreds of thousands or millions of panels
  - Will never be able to (or want to) view all of them!

Scaling Trellis

- Scaling Trellis:
  - Data are split into meaningful subsets, usually conditioning on variables of the dataset
  - A visualization method is applied to each subset
  - Panels are arranged in an array of rows, columns, and pages, resembling a garden trellis

Scaling Trellis with Cognostics

- Scaling Trellis:
  - Data are split into meaningful subsets, usually conditioning on variables of the dataset
  - A visualization method is applied to each subset
  - A set of cognostics that measure attributes of interest for each subset is computed
  - Panels are arranged in an array of rows, columns, and pages, resembling a garden trellis, with the arrangement being specified through interactions with the cognostics
- Can be achieved with the `autocogs` and `trelliscopejs` R packages

Calculated Cognostics

```
housing %>%
  group_by(county, state) %>%
  nest() %>%
   mutate(
    loess = lapply(data, function(dt) {
      loess(
        medListPriceSqft ~ as.numeric(time),
        data = filter(dt, !is.na(medListPriceSqft))
      )
    })
) %>%
mutate(
  res_std_err = map_dbl(loess, ~ cog(.$s, "residual standard error")),
  enp = map_dbl(loess, ~ cog(.$enp, "effective number of parameters")),
  mean_med_price = map_dbl(data,
    ~ cog(mean(.$medListPriceSqft), "mean of median list price”)),
  n_obs_list = map_dbl(data,
    ~ cog(sum(!is.na(.$medListPriceSqft)), "number of observations”)),
  zillow_href = map2(county, state, function(county_, state_) {
    cog_href(
      sprintf("http://www.zillow.com/homes/%s-%s_rb/", county_, state_),
      "zillow link”
    )})
)
```

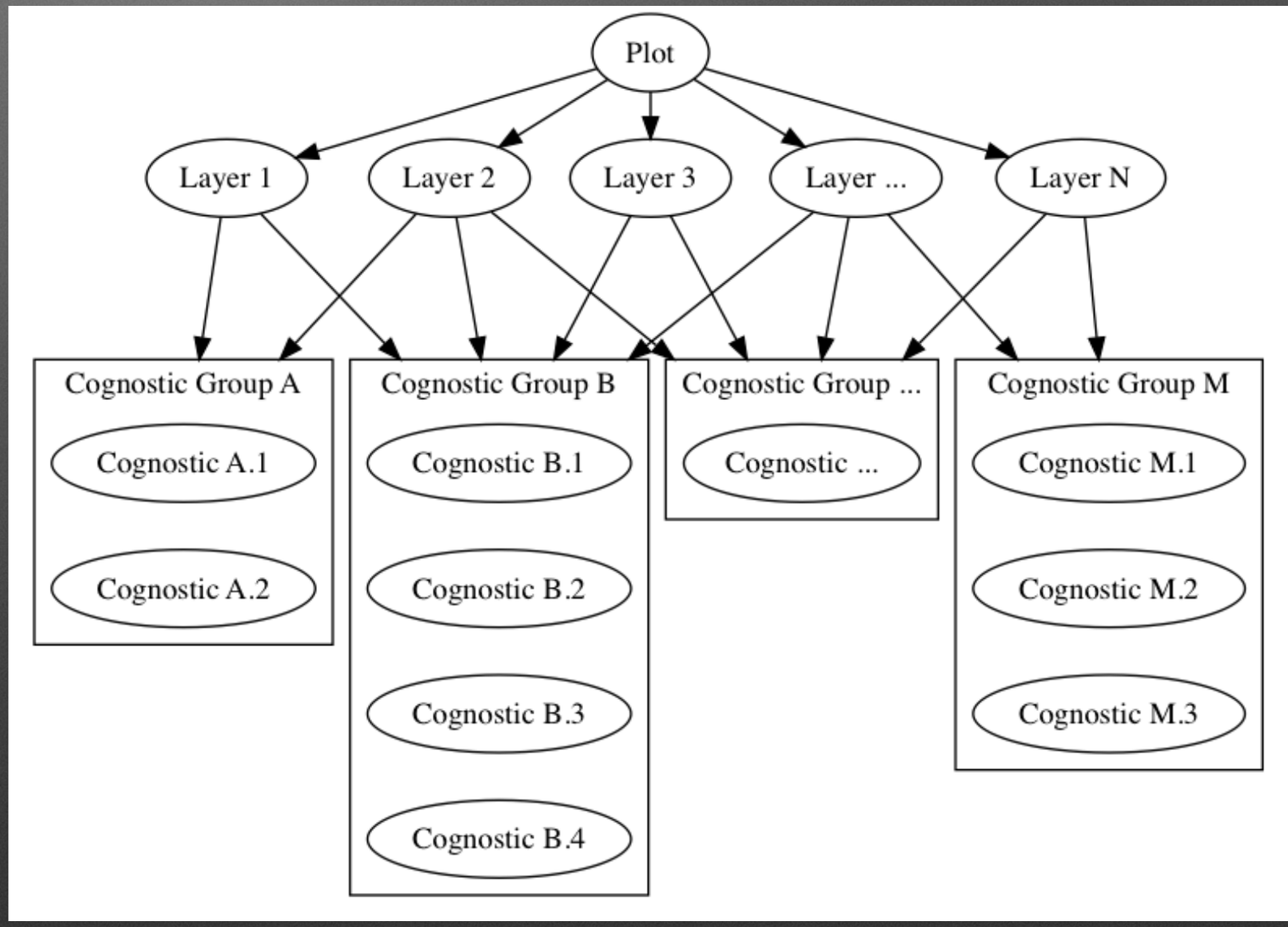
Trelliscope Demo

Automatic Cognostics

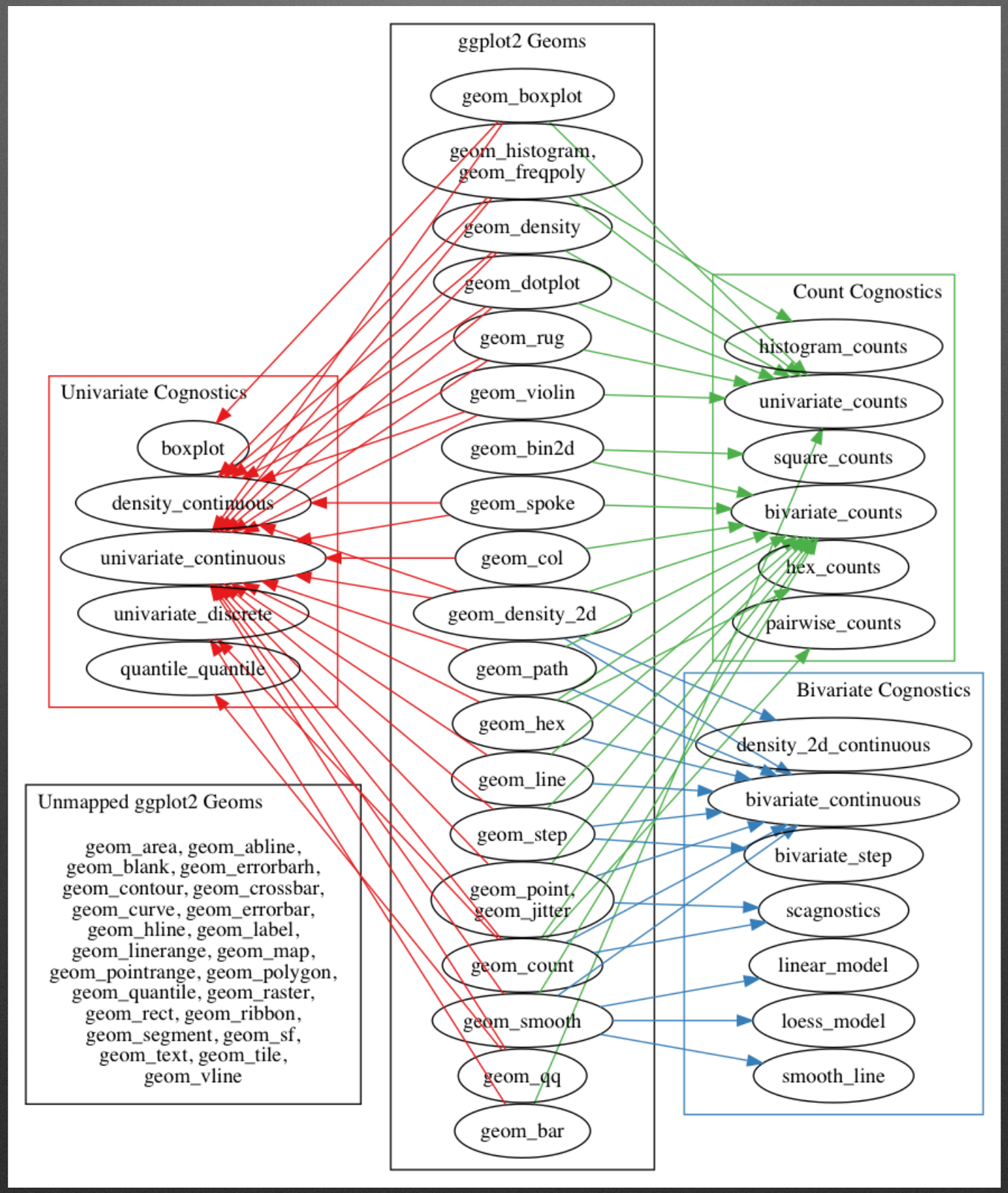
- Cumbersome to manually specify many cognostics for a Trelliscope display
- Should be able to automatically compute cognostics based on the context of what is being plotted
  - Help foster a scalable Trellis system
- Analyze the plot objects and choose "best" cognostics based on the plot specification

Plot Layers

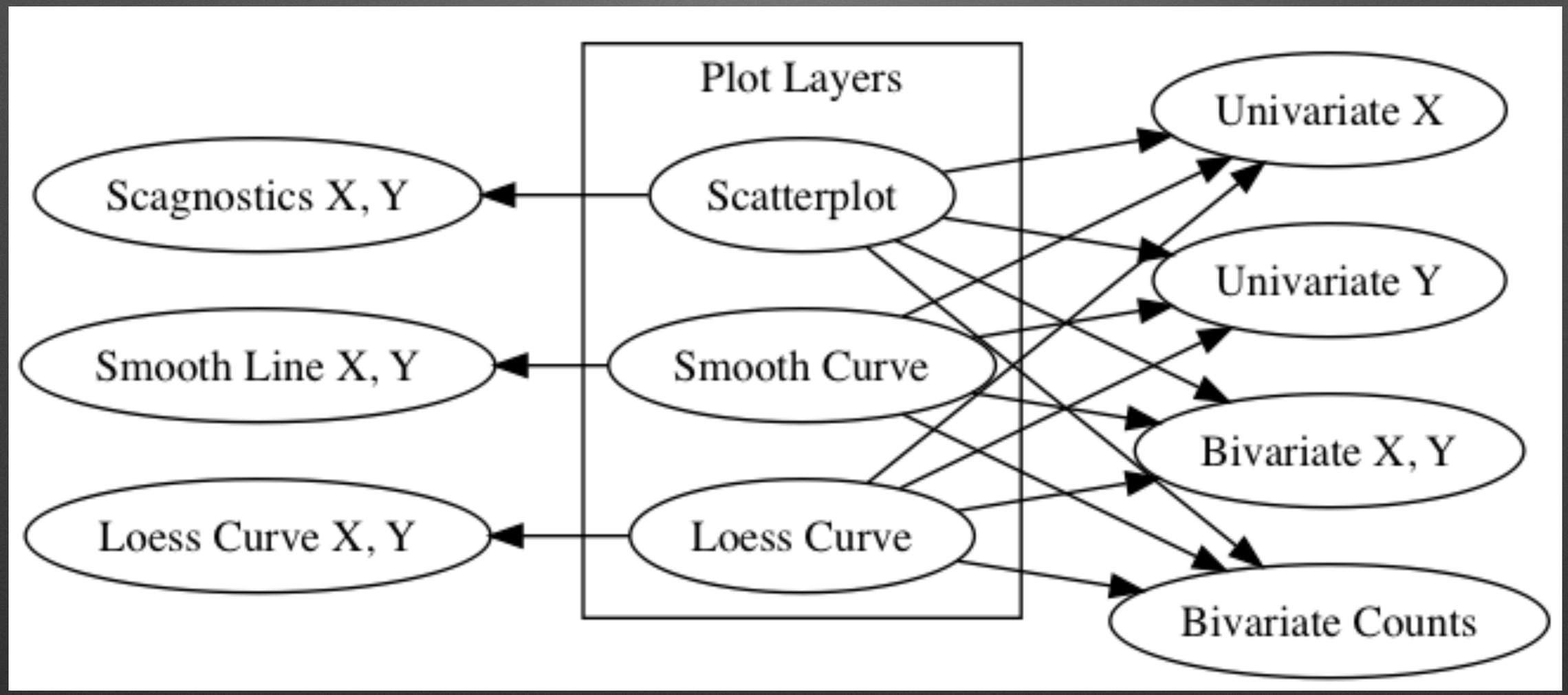
- Each plot may contain many layers
- Each layer may produce many cognostics
- Many to many mapping
  - Only need unique cognostic groups



ggplot2 Layer to Cognostic Mapping



Housing Example



Housing Cognostics

- Univariate X:
  `minimum, maximum, mean, median, variance`
- Univariate Y
  `minimum, maximum, mean, median, variance`
- Bivariate X, Y:
  `covariance, correlation`
- Bivariate Counts X, Y:
  `n, n_both_NA, n_or_NA, n_x_na, n_y_na`
- Scagnostics X, Y:
  `Outlying, Skewed, Clumpy, Sparse, Striated, Convex, Skinny, Stringy, Monotonic`
- Smooth Curve X, Y:
  `mse, max_deviation, max_deviation_location`
- Loess Curve X, Y:
  `enp, s, trace.hat, span, degree, iterations`

35 Cognostics!

Application

```
housing_y <- range(housing$medListPriceSqft, na.rm = TRUE)

# set up data
housing %>%
  group_by(county, state) %>%
  nest() %>%
  filter(map_dbl(data, nrow) > 10) %>%

  # visualize data
  mutate(
    panel = map_plot(data, ~
      ggplot(.x, mapping = aes(time, medListPriceSqft)) +
        geom_point() +
        geom_smooth(method = "loess") +
        ylim(housing_y[1], housing_y[2]))) %>%



# create trelliscope application
trelliscope(
  "all_states", "housing", nrow = 3, ncol = 5,
  path = "housing_list_price",
  state = list(sort =
    list(sort_spec("county"), sort_spec("state"))))
```

Application

```
housing_y <- range(housing$medListPriceSqft, na.rm = TRUE)

# set up data
housing %>%
  group_by(county, state) %>%
  nest() %>%
  filter(map_dbl(data, nrow) > 10) %>%

  # visualize data
  mutate(
    panel = map_plot(data, ~
      ggplot(.x, mapping = aes(time, medListPriceSqft)) +
        geom_point() +
        geom_smooth(method = "loess") +
        ylim(housing_y[1], housing_y[2]))) %>%

# add automatic cognostics
add_panel_cogs() %>%

# create trelliscope application
trelliscope(
  "all_states_auto", "housing", nrow = 3, ncol = 5,
  path = "housing_list_price_auto",
  state = list(sort =
    list(sort_spec("county"), sort_spec("state"))))
```

Application Output
No Cognostics

```
# A tibble: 2,945 x 4
                  county  state                 data     panel
                  <fctr> <fctr>               <list>    <list>
 1 Los Angeles County      CA <tibble [92 x 5]> <S3: gg>
 2         Cook County      IL <tibble [97 x 5]> <S3: gg>
 3     Maricopa County      AZ <tibble [97 x 5]> <S3: gg>
 4    San Diego County      CA <tibble [97 x 5]> <S3: gg>
 5       Orange County      CA <tibble [92 x 5]> <S3: gg>
 6        Kings County      NY <tibble [97 x 5]> <S3: gg>
 7  Miami-Dade County      FL <tibble [97 x 5]> <S3: gg>
 8       Dallas County      TX <tibble [97 x 5]> <S3: gg>
 9       Queens County      NY <tibble [97 x 5]> <S3: gg>
10    Riverside County      CA <tibble [97 x 5]> <S3: gg>
# ... with 2,935 more rows
```

Application Output With Cognostics

```
# A tibble: 2,945 x 11
                county  state              data    panel    `_scagnostic`
                <fctr> <fctr>            <list>   <list>           <list>
 1 Los Angeles County      CA <tibble [92 x 5]> <S3: gg> <tibble [1 x 9]>
 2        Cook County      IL <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
 3    Maricopa County      AZ <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
 4   San Diego County      CA <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
 5      Orange County      CA <tibble [92 x 5]> <S3: gg> <tibble [1 x 9]>
 6       Kings County      NY <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
 7  Miami-Dade County      FL <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
 8      Dallas County      TX <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
 9      Queens County      NY <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
10   Riverside County      CA <tibble [97 x 5]> <S3: gg> <tibble [1 x 9]>
# ... with 2,935 more rows, and 6 more variables: `_x` <list>, `_y` <list>,
#   `_bivar` <list>, `_smooth` <list>, `_loess` <list>, `_n` <list>
```

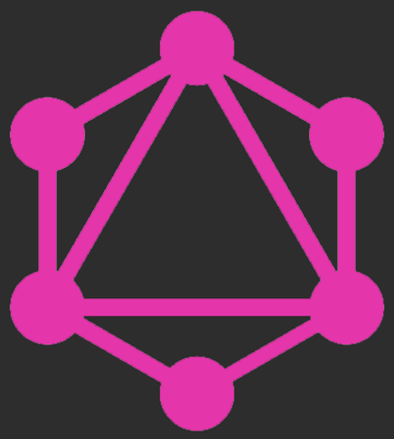
`facet_trelliscope(~ a + b)`

- Create a full trelliscopejs application using ggplot2 syntax

```
ggplot(housing, mapping = aes(time, medListPriceSqft)) +
  geom_point() +
  geom_smooth(method = "loess") +
  facet_trelliscope(~ county + state, auto_cog = TRUE)
```

Automatic Cognostics

- Generate using existing panels
- Multiple cognostic groups for each layer
- Framework built for any layered plotting library

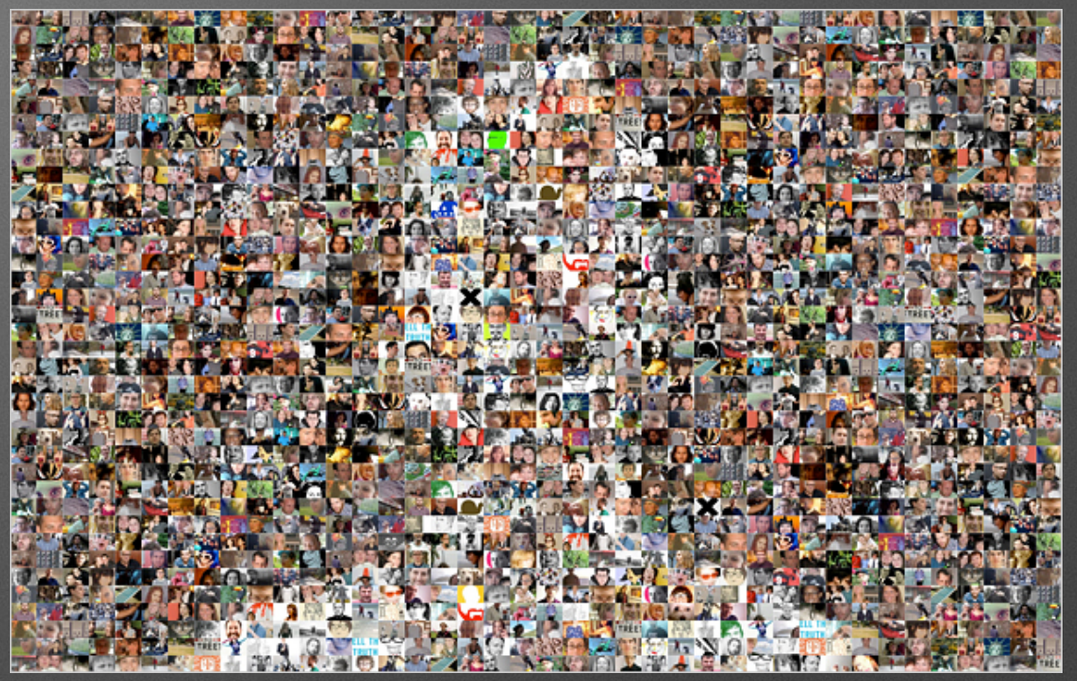


gqlr = GraphQL + R

Data Query API + R

NSF Grant: DGE-1333468

Example: Facebook Friend Info



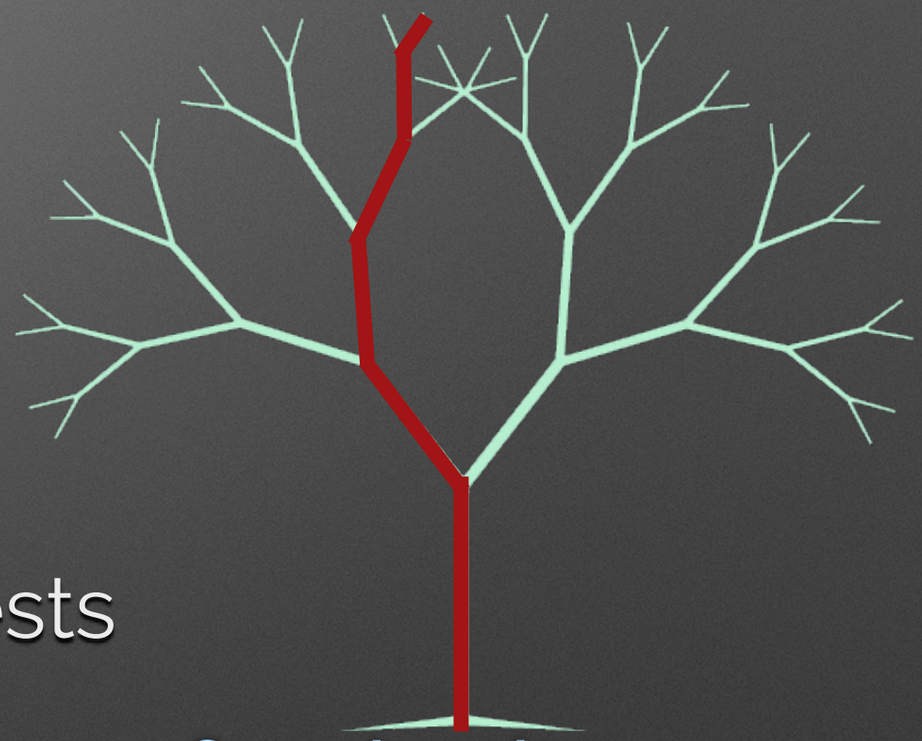
- Display all of my friends'
  - profile picture
  - full name

- REST (naive server setup)
  - `Ask for all n friend IDs`
  - `For each friend ID:`
    - `Ask server for friend ID's profile information`

- Total query count… **`1 + n`**

Facebook Friend Info Query Limitations

- **n + 1** queries!
  - Browsers limited to **~10** parallel connections per host
  - 1000 friends
    - ~10 seconds to load 1001 requests at 0.1 s/request
  - only **one** part of the website!
- Bottleneck is with the data server API

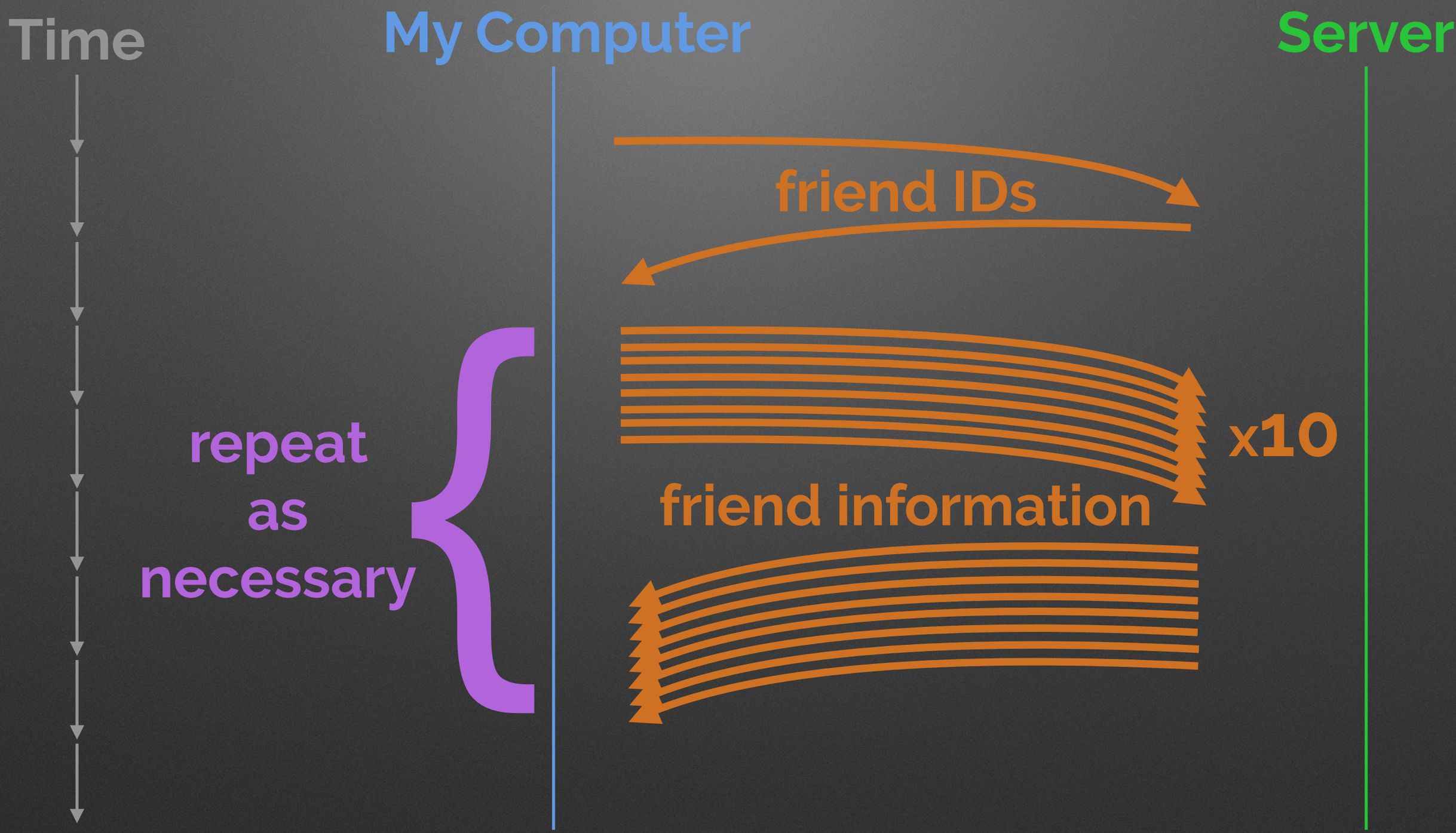


http://www.browserscope.org/?category=network

Data Server API Spectrum

- Naive REST
  - Easy to implement
  - **Very slow** to execute (n + 1 queries)

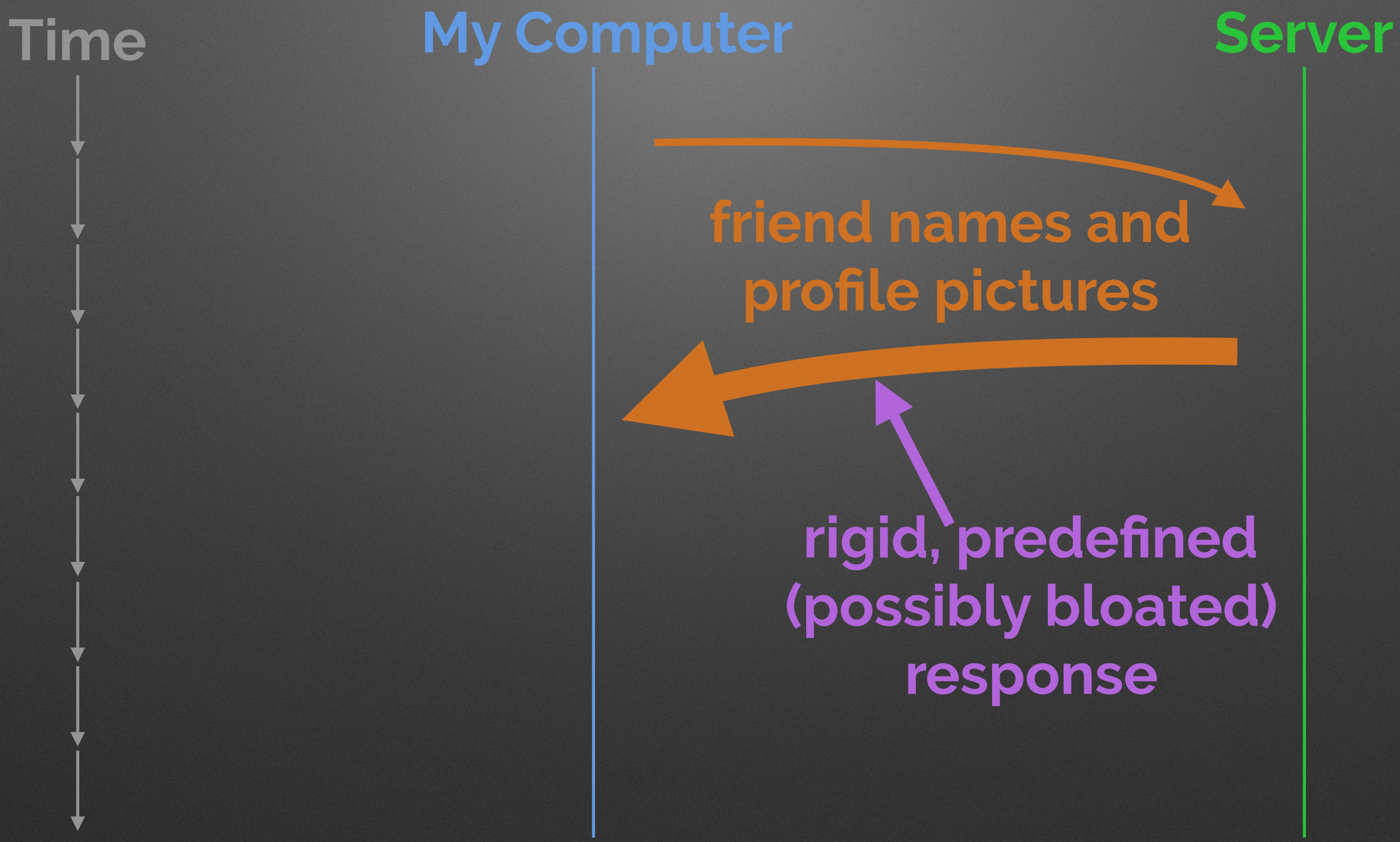
Naive REST


Time
My Computer
Server
friend IDs
repeat
as
necessary
x10
friend information

Data Server API Spectrum

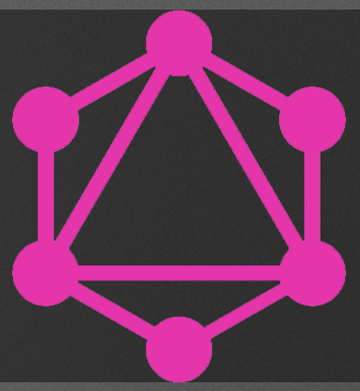
- Naive REST
  - Easy to implement
  - **Very slow** to execute (n + 1 queries)
- Custom Server
  - Difficult to implement
  - **Fast** (1 query)
  - Every query requirement needs a custom server response
  - No separation of browser information needs and server information availability
    - Typically causes over-fetching of data

Custom Server



Naive + Custom Data Server API?

GraphQL



- **Graph Q**uery **L**anguage
  - “A data query language for your API”
- Facebook open sourced the specification in mid 2015
- Backend agnostic data query language built upon strong-typed hierarchical sets of fields.
  - "strong type system" is described as one in which there is no possibility of an unchecked runtime type error
- “The query is shaped just like the data it returns. It is a natural way for product engineers to describe data requirements.”
  - Non-rigid
  - Avoids under-fetching and over-fetching

http://graphql.org/

Two parts

- Schema
  - Defines the strong typed objects
- Query
  - Asks for objects and fields defined in the Schema

Facebook Example: GraphQL

- Schema

```
scalar LocalURI

type User {
  id: Int
  name: String
  profPic: LocalURI
  friends: [User]
}

type Query {
  user(id: Int!): User
}
```

- Query

```
query friends_info {
  user(id: 3945) {
    name,
    profPic
    friends {
      id,
      name,
      profPic
    }
  }
}
```

Facebook Example: Result


```
· {
    “user”: {
      “name”: “Barret”,
      “profPic”: “/p/3945”,
      “friends”: [
{“id”: 1436, “name”: “Tracy”,  “profPic”: “/p/1436”},
{“id”: 3849, “name”: “Gui”,    “profPic”: “/p/3849”},
{“id”: 5978, “name”: “Jiasen”, “profPic”: “/p/5978”},
{“id”: 9632, “name”: “Ayu”,    “profPic”: “/p/9632”},
{“id”: 2931, “name”: “Emery”,  “profPic”: “/p/2931”},
…
    ]
  }
}
```

Endless Query Options

- Only restricted by Schema definition
  - User's `name` only
  - User's `name` and `profPic`
  - User's friends of friends' `id` and `profPic`

gqlr = GraphQL + R

- GraphQL with the power of R
  - Endless query options with GraphQL
  - Fast iteration speed of R
  - Many external libraries may be leveraged for computation
- Retrieve data from...
  - memory / disk
  - external databases (hadoop, mysql, ...)
  - **simulation / calculation**
    - Use any R package or personal scripts!

Power of R

- type User {
    id: Int
    name: String
    profPic: LocalUrl
    friends: [User]
    bestFriends: [User]
  }
- ‘bestFriends’ should be calculated on the fly
  - Expensive calculation to do for **everyone** at **all times**
  - fastcluster::hclust
    - External package!

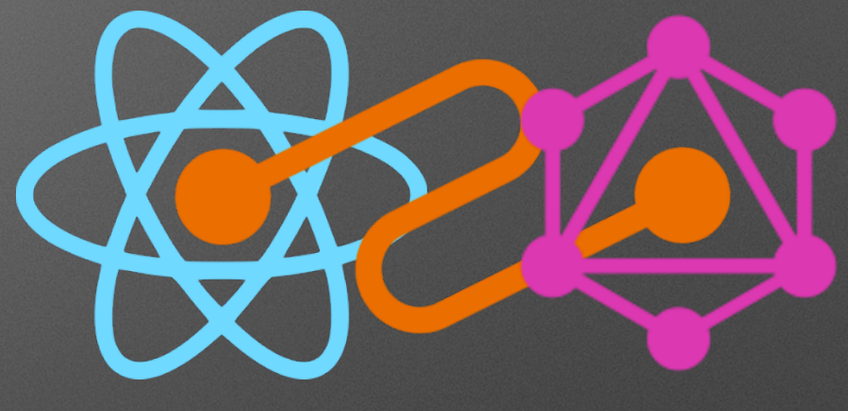
`gqlr`
Native R Implementation

- Specification
  - 133 printed pages
  - Defines implementation rules
- Built using R6
  - Allows for non-standard R interactions

```
n$rand
#> [1] 0.2648
n$rand
#> [1] 2.171
```

Immediate Uses

- relay web applications
  - https://facebook.github.io/relay/
- ex: trelliscopejs
  - Complex R application
  - Migrated from shiny to pure javascript using React
  - Data could be retrieved with Relay
  - https://hafen.github.io/trelliscopejs



Websites

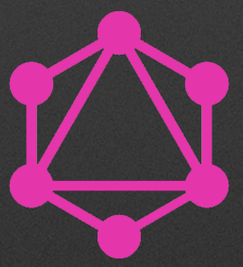
- Main GraphQL Website
  - graphql.org
- Specification Document
  - facebook.github.io/graphql
- Javascript Implementation of GraphQL
  - github.com/graphql/graphql-js
- Learn GraphQL
  - github.com/dwyl/learn-graphQL

gqlr = GraphQL + R

- GraphQL
  - Reduce required data queries
  - Eliminate under and over fetching of data
- R
  - Maintain fast iteration speed
  - Utilize 3rd party packages

Not Too Distant Future

- Large data hosted in Hadoop
  - Conditioned into smaller subsets
- Execute a linear model for all subsets
- Display the diagnostics using `GGally::ggnostic`
- Automatically calculate all cognostics of `ggnostic` display
- Traverse diagnostic displays within `trelliscopejs` communicating with `gqlr`



Generalized Plot Matrices, Automatic Cognostics, and Efficient Data Exploration

Thank You!

- Committee
  - Dr. William Cleveland
  - Dr. Ryan Hafen
  - Dr. Bowei Xi
  - Dr. Vinayak Rao

```
enum Topic {GGDUO, AUTOCOGS, GQLR}
type Answer {
  id: Int
  question: String
  answer: String
  confidence: Float
}
type Query {
  question(id: Int!, topic: Topic): Answer
}
schema {
  query: Query
}
```