Teaching and Educational Methods

Gold in Them Tha-R Hills: A Review of R Packages for Exploratory Data Analysis

Kota Minegishi[a] and Taro Mieno[b]
[a] University of Minnesota, Twin Cities, [b] University of Nebraska-Lincoln

JEL Codes: A2, Q1, Y1
Keywords: Exploratory data analysis, data science, data visualization, R programming

Abstract
With an accelerated pace of data accumulation in the economy, there is a growing need for data literacy and practical skills to make use of data in the workforce. Applied economics programs have an important role to play in training students in those areas. Teaching tools of data exploration and visualization, also known as exploratory data analysis (EDA), would be a timely addition to existing curriculums. It would also present a new opportunity to engage students through hands-on exercises using real-world data in ways that differ from exercises in statistics. In this article, we review recent developments in the EDA toolkit for statistical computing freeware R, focusing on the tidy verse package. Our contributions are three-fold; we present this new generation of tools with a focus on its syntax structure; our examples show how one can use public data of the U.S. Census of Agriculture for data exploration; and we highlight the practical value of EDA in handling data, uncovering insights, and communicating key aspects of the data.

1 Introduction

There's gold in them thar hills! —Mark Twain in The American Claimant

A hundred seventy years ago Americans flocked to California in search of gold. The Gold Rush left the country with a powerful image of massive realignment of capital and labor in search of new economic opportunities. With each subsequent era came new manifestations of the Gold Rush in the form of booming industries, invoking a sense of new, ground-breaking opportunities that could lead to permanent structural change in the existing business environments. Today, businesses are gathering and accumulating an enormous amount of data: effective goldmines. In this new Gold Rush, the demand for the skills to understand, explore, and apply data is accelerating. Computer programmers and data scientists are particularly in high demand, and their tool kit is expanding rapidly. In preparing students for an increasingly data-driven world, applied economics programs have an increased role to play through teaching data literacy and modern data analytics skills.

A good starting point may be to teach relevant tools of data exploration and visualization, also known as exploratory data analysis (EDA), that are popular in the field of data science. The exploratory nature of EDA contrasts with statistical modeling and hypothesis testing, a long-standing tradition in modern economics curriculums. An increasing number of economics courses integrate statistical programming in R as an integral topic. Current examples include Microeconomics with R by John Humphries at Yale University, Methodology of Economic Research by Jude Bayham at Colorado State University, econometrics course materials taught with R by Ed Rubin, Data Science for Economists by Grant

McDermott at University of Oregon, and Applied Econometrics by Taro Mieno at University of Nebraska–Lincoln as far as the authors are aware of. Indeed, the tools of EDA are generally complementary to the teaching of analytical skills and thought processes emphasized in applied economics. Teaching EDA tools would be not only timely but also stimulating for students who have an interest in learning to use real-world data on current socioeconomic issues. Hands-on EDA exercises can provide a vital opportunity for students to acquire practical data analysis skills beyond the usual exercises in statistics.

In this article, we review recent developments in the EDA toolkit in statistical computing freeware R. Our intended audience includes course instructors, graduate students, and advanced undergraduate students particularly those who are pursuing independent studies, participating in research projects, or serving as teaching assistants. We use data sets familiar to agricultural economists for illustration. Our contributions are three-fold: we present this new generation of tools with a focus on its syntax structure, our examples show how one can use public data of the U.S. Census of Agriculture for data exploration, and we highlight the practical value of EDA in handling data, uncovering insights, and communicating key aspects of the data. Our review focuses on the tools of the `tidyverse` package, a meta package that includes *ggplot2* and *dplyr* and uses a streamlined coding syntax across its member packages (Wickham et al. 2019).[1] In writing this article, we borrow core concepts from **R for Data Science** (Wickham and Grolemund 2017). For interested readers, additional resources include **ModernDive** (Ismay and Kim 2019), **Data Visualization with R** (Kabacoff 2018), **Data Visualization: Practical Introduction** (Healy 2018) and **Geocomputation with R** (Lovelace, Nowosad, and Muenchow 2019).[2] All R code used in this document is made available in the supplementary appendix.[3]

The rest of the article is organized as follows. We provide a short, general comparison between R and Stata, a popular proprietary statistical software among economists. The main contents of our review of R tools consist of four sections that (a) introduce core data visualization methods of *ggplot2*, (b) demonstrate the application of data transformation methods of *dplyr* with U.S. agriculture data, (c) provide an analytical example within a data exploration narrative, and (d) briefly describe additional tools. The final section concludes the article.

2 Comparison of R and Stata

As a general comparison, we comment on the relative strengths and weakness of two commonly used software programming languages in the field of economics, R and Stata.[4]

2.1 A Basic Introduction

R, formally known as *R Projects*, is a statistical computing, graphics, and programming language that is available free of charge. R is not managed by a single person or company but instead by an "R core group."[5] The R core group has the authority to modify the R source code archive. For most users, it suffices to know that R simply executes commands according to programs, or R functions, that are loaded by default and by the user. To execute commands beyond basic computations and visualization tasks, R users need to load R packages, collections of R functions developed and shared by other R users. Which packages to use depends

[1] They are not part of the base package. To install a R package, execute the code in the R console, for example: `install.packages("tidyverse")`.

[2] R for Data Science: https://r4ds.had.co.nz/, ModernDive: https://moderndive.com/, Data Visualization with R: https://rkabacoff.github.io/datavis/, Data Visualization A Practical Introduction: http://socviz.co/index.html, Geocomputation with R: https://geocompr.robinlovelace.net/.

[3] https://github.com/tmieno2/R-AETR

[4] Software download: https://cloud.r-project.org/ and https://download.stata.com/download/.

[5] https://www.r-project.org/contributors.html.

on the user's objectives and personal preferences. For example, two popular EDA toolboxes are the *tidyverse* package, which is our focus in this article, and the *data.table* package.

Stata is a proprietary statistics software from StataCorp. In most universities, students can access Stata in their computer labs through a site license. As of December 2019, the Stata perpetual license for U.S. students is $225 for Stata/IC (the least powerful version), $425 for Stata/SE, $595 for Stata/MP 2-core (midrange capabilities), and $795 for Stata/MP (the most powerful). Short-term U.S. student licenses are also available for $48 for Stata/IC and $125 for Stata/SE for 6 months. StataCorp is responsible for software descriptions, updates, and additions of Stata commands. Separately, some user-contributed Stata packages, a collection of Stata *ado* files, are available through RePEc (which stands for Research Papers in Economics). Also, StataCorp maintains a quarterly publication of the Stata journal for user-contributed statistical techniques and effective teaching methods using Stata.

2.2 Statistical Capability

R is open-source software with a rapidly expanding toolkit built by the R user community across diverse fields of statistics and sciences. The R toolkit includes advanced tools of machine learning, Bayesian statistics, and spatial statistics that are of interest to many economists, as well as statistical tools in other disciplines like biostatistics that may help economists working on interdisciplinary research. R offers rich tools in some fields of econometrics, including, for example, linear or quadratic programming (*Rglpk* and *ipotr* packages), nonlinear optimization (*nloptr* package), and advanced quantile regression analyses (*quantreg*, *quantreg.nonpar*, and *bayesQR* packages).

Stata's development of new tools primarily rests on StataCorp's undertaking. Given its limited resources, the company focuses on tools for social scientists, including economists. For instance, Stata offers a variety of estimation options for state-of-the-art treatment effects and panel data estimation techniques that are useful to economists. Advanced coding implementation of customized nonlinear estimation is also available.[6] The documentation of various commands in Stata is consistently managed by the company and hence user-friendly; in contrast the user-contributed projects of R may lack consistent documentation or transferable command syntaxes across various packages. Thus, a familiarity with both R and Stata would give the user access to a wide range of statistical methods, some of which may be available in one software but not in the other.

2.3 Machine Learning Methods

There is a growing interest in R among agricultural economists, and it can be explained by the increased importance of Big Data and the expanding capabilities of machine learning methods (Coble et al. 2018; Storm, Baylis, and Heckelei 2019). Numerous packages that implement state-of-the-art machine learning methods are available in R, including LASSO, Random Forest, Neural Network, and Boosted Regression. The *keras* and *tensorflow* packages handle Convolutional Neural Network (CNN), a workhorse for image processing used in facial recognition and autonomous driving. An interesting application of CNN may include spatial data analysis (Storm, Baylis, and Heckelei 2019). The *rnn* package allows for recurrent neural network modeling, which is particularly suitable for state-dependent time-series analysis and a certain type of price analysis. The *grf* package leads the generalized random forest framework, which includes causal forest, quantile forest, and instrumental forest developed by Athey, Tibshirani, and Wager

6 https://blog.stata.com/2016/01/26/programming-an-estimation-command-in-stata-a-review-of-nonlinear-optimization-using-mata/

(2019). The *XGBoost* package offers extreme gradient boosting regression, which has been shown to outperform other machine learning methods in many applications.

In the latest version of Stata 16, StataCorp has introduced LASSO commands. In addition, user-contributed packages such as *LASSOPACK* (LASSO, elastic net, and ridge regressions), *RFOREST* (random forest classification and regression), and *KFOLDCLASS* (K-fold cross-validation for binary outcomes) are available. It is plausible that many machine learning algorithms will be gradually made available.

2.4 Spatial Data Handling

Many data analyses in agricultural economics involve spatial considerations. R offers an extensive capability in processing spatial data (*sp*, *sf*, *raster*, *rgdal*, and *rgeos* packages are some examples) and creating geographical maps (*ggplot2* and *tmap* packages have wide use). If for instance, one is interested in understanding the impact of climate on cropping patterns at the sub-county level, he or she could combine the Cropland Data Layer (CDL) files and the county boundaries data to summarize a mixture of cropping patterns for each county, all of which can be done within R without having to use specialized programs such as ArcGIS or QGIS.[7] In contrast, Stata has a very limited capability for handling spatial data or generating geographic data figures. One exception may be the user-contributed mapping commands like *spmap* and *maptile*.

2.5 Publicly Available Data

Recent developments in R include packages that are dedicated specifically for downloading publicly accessible data. One can download data from the USDA NASS CDL (*cdlTools* package), USGS and EPA hydrologic and water quality data (dataRetrieval), Quick Stats (*rnassqs* package), PRISM (*prism* package), Daymet (*daymetr* package), Sentinel-2 satellite imagery data (sen2r package), the National Elevation Data Set digital elevation models, the NCSS Soil Survey Geographic data set, and many others (*FedData* package). These R packages can automate the process of manually downloading individual public data files. Additionally, the *httr* package allows for data requests via Application Programming Interface (API), and the *jsonlite* package helps process JSON data files that are common in API outputs. Stata has a capability to utilize API through the *winexec curl* command. Also, downloaded data in XML or JSON format can be imported into Stata via *xmluse* or *insheetjson*, respectively.

3 Data Visualization with *ggplot2*

This section highlights simple data visualization methods with R's *ggplot2* package for creating scatter, line, and bar plots.[8] The *ggplot2* syntax has three essential components for generating data plots: *data*, *aes*, and *geom*. It implements the following philosophy:

A statistical graphic is a mapping of ***data*** *variables to* ***aesthetic*** *attributes of* ***geometric*** *objects.*
(Wilkinson 2005, p. 42)

where the data, aesthetic attributes, and geometric objects are programmed as follows:

- *data*: a data frame; e.g., the first argument in *ggplot(data, ...)*.

[7] For example, see R as GIS for Economists: https://tmieno2.github.io/R-as-GIS-for-Economists/.

[8] For basic R tutorials, try http://www.cookbook-r.com/ or https://en.wikibooks.org/wiki/R_Programming/Sample_Session. A useful material for teaching may be https://psyteachr.github.io/.

- *aes*: *x* and *y* variables specifying the horizontal and vertical axes and other variables by which data can appear in different colors, shapes, sizes, etc.; e.g., *aes(*x = *var_*x*, y* = *var_*y*, color = var_*z*)*.
- *geom*: geometric objects such as points, lines, bars, etc.; e.g., *geom_point()*, *geom_line()*, *geom_bar()*, *geom_histogram()*.

This simple philosophy provides an easy way for remembering how to relate the three components with each other in coding. Note that data sets are often referred to as data frames, corresponding to R's *data.frame* class objects that, unlike matrix class objects, can contain both string and numeric variables in columns.

We now examine some basic examples. The following code produces scatterplots of horsepower and miles per gallon using the *mtcars* data set, a sample data set automatically loaded in *base R* (Figure 1). It came from the 1974 Motor Trend U.S. magazine and contains 11 automobile specification attributes for 32 cars, including attributes like gross horsepower (hp), miles per gallon (mpg), number of cylinders (cyl), automatic transmission indicator (am), and weight in 1,000 of pounds (wt).[9]

```r
# scatterplot of hp and mpg
ggplot(mtcars, mapping = aes(x = hp, y = mpg)) +
  geom_point()

# convert variable cylinder into a categorical variable
mtcars$cyl <- as.factor(mtcars$cyl)

# scatterplot with added color and shape by cylinder
ggplot(mtcars, mapping = aes(x = hp, y = mpg, color = cyl)) +
  geom_point(aes(shape = cyl))
```

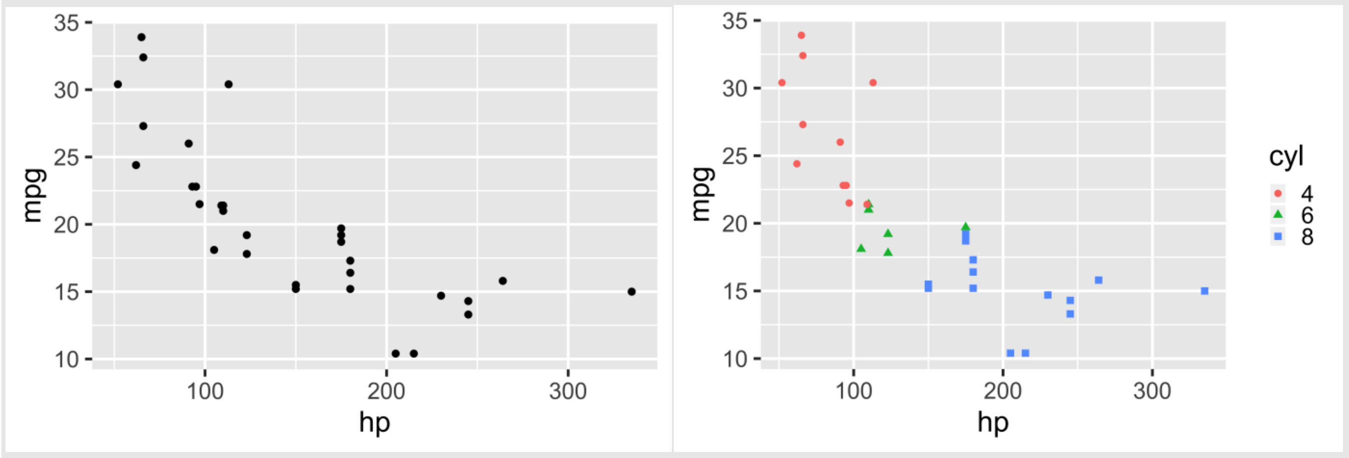


Figure 1. Example of Scatterplots Using the mtcars Data Set in Base R

In the next example, we add more layers of geometric objects, see bullet point "geom" above (Figure 2). By default, a geometric object inherits the aesthetic attributes specified in *gglot(data, aes())*. To change those attributes, one needs to provide specific attributes for each geometric object. In the first two plots, note that the presence or absence of a color attribute specification in *ggplot(data, aes())*, which implies different color attribute specifications in *geom_smooth()*. The third plot contains an example of fixed aesthetic attributes like color and point size that are specified outside *aes() and hence* do not depend on

[9] While unrelated to agriculture, this data set is commonly used for basic R tutorials and hence good to be familiar with.

the data. Also, one can add a geometric object with a new data set. For example, the third plot contains a geometric object based on a subset of the data.

```r
# add a layer of linear regression model fit for each cylinder type
ggplot(mtcars, aes(x = hp, y = mpg, color = cyl)) +
  geom_point(aes(shape = cyl)) +
  geom_smooth(method = lm)

# add a layer of smooth regression fit (locally estimated scatterplot
smoothing: loess) across all cylinder types
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  geom_smooth()

# add a layer of large yellow dots to indicate automatic transmission
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(data = filter(mtcars, am == 0), color = "yellow", size = 5) +
  geom_point(aes(shape = cyl, color = cyl)) +
  geom_smooth()
```

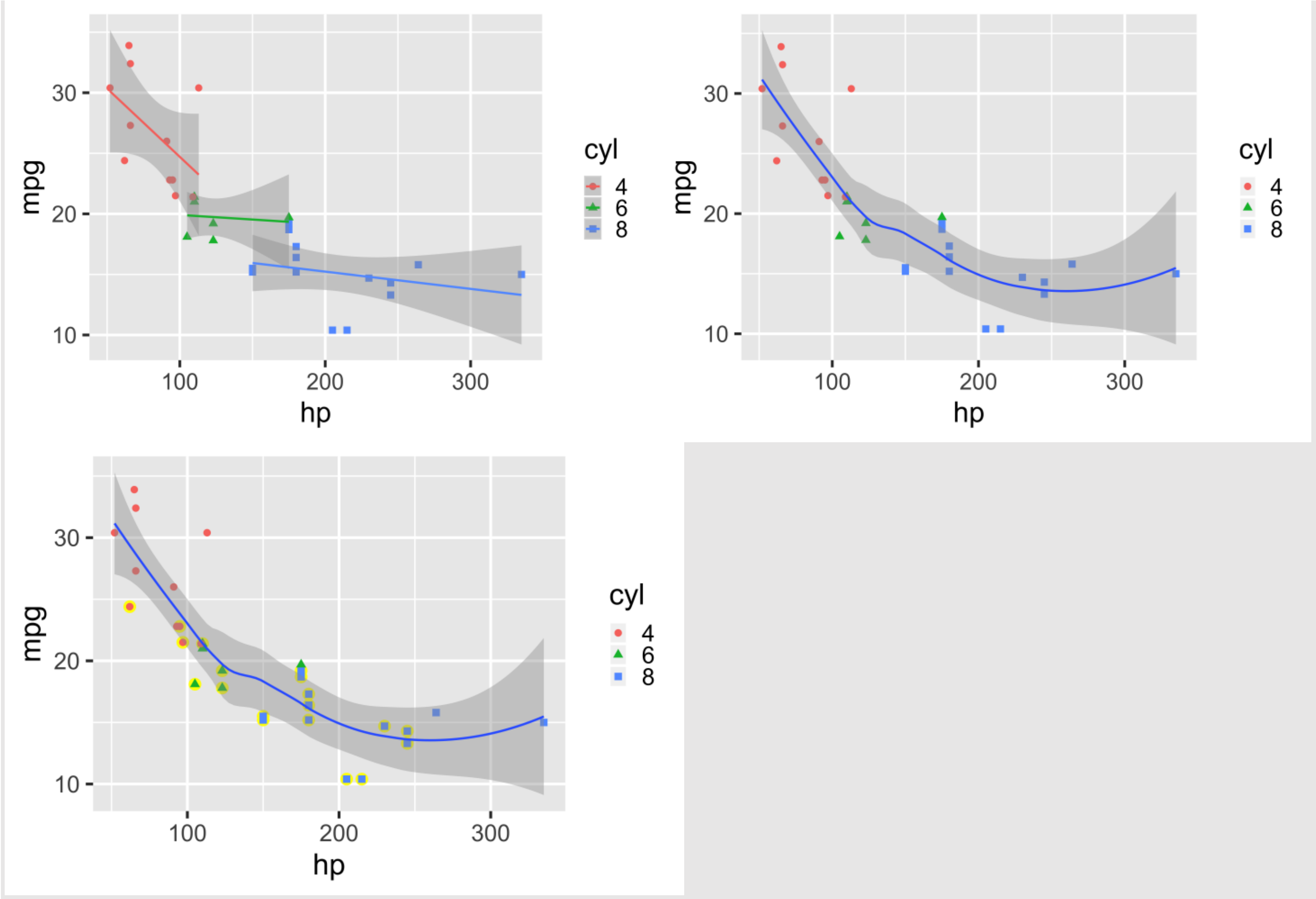


Figure 2. Example of Scatterplots with Linear Model and Smooth Fits Using the mtcars Data

Additionally, a *facet_wrap()* or *facet_grid()* layer splits the data into subsets by a categorical variable(s) and generates multiple data plots on those subsets (Figure 3).

```r
# add a character variable for transimission type
mtcars$am_char <- recode(c(mtcars$am), "0" = "automatic", "1" = "manual")

# plot subsets of data by transmission type
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  facet_wrap( ~ am_char)

#  plot subsets of data by transmission type and number of gears
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  facet_grid(gear ~ am_char)
```

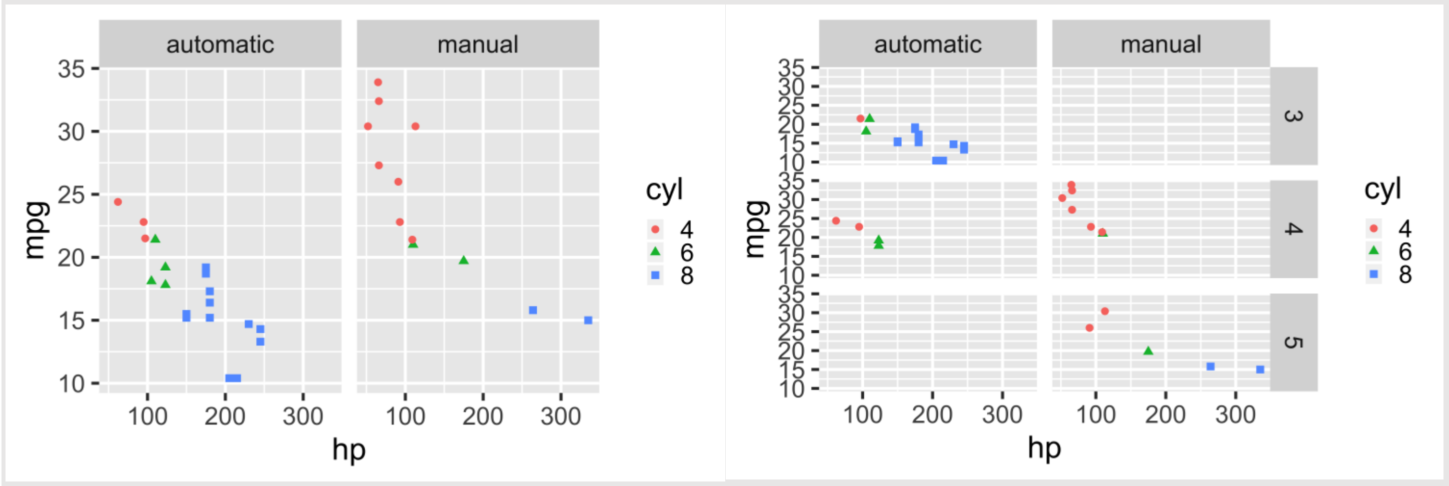


Figure 3. Example of Scatterplots for Subsets of the mtcars Data
Note: The data are split into two subsets by transmission type (top) and six subsets by the combination of transmission type and number of cylinders (bottom). Variables mpg, hp, and cyl refer to miles per gallon, horse power, and the number of cylinders, respectively.

Various cosmetic adjustments can be controlled through additional layers of coordinate attributes (scale and coord) and other graphics attributes (labs, theme, and guides) as demonstrated in Figure 4.

```r
# change the displayed values on the y axis
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  scale_y_continuous(breaks = seq(10, 36, by = 4))

# map in log10 scale
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  scale_x_log10() + scale_y_log10()

# change theme to black and white and overwrite axis labels
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  theme_bw() + labs(x = "Horse power", y = "Miles per gallon")

# overwrite the *joint legend* for color and shape attributes
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  guides(
    color = guide_legend(title ="cylinder", override.aes = list(size = 4)),
    shape = guide_legend(title ="cylinder", override.aes = list(size = 4))
    )
```

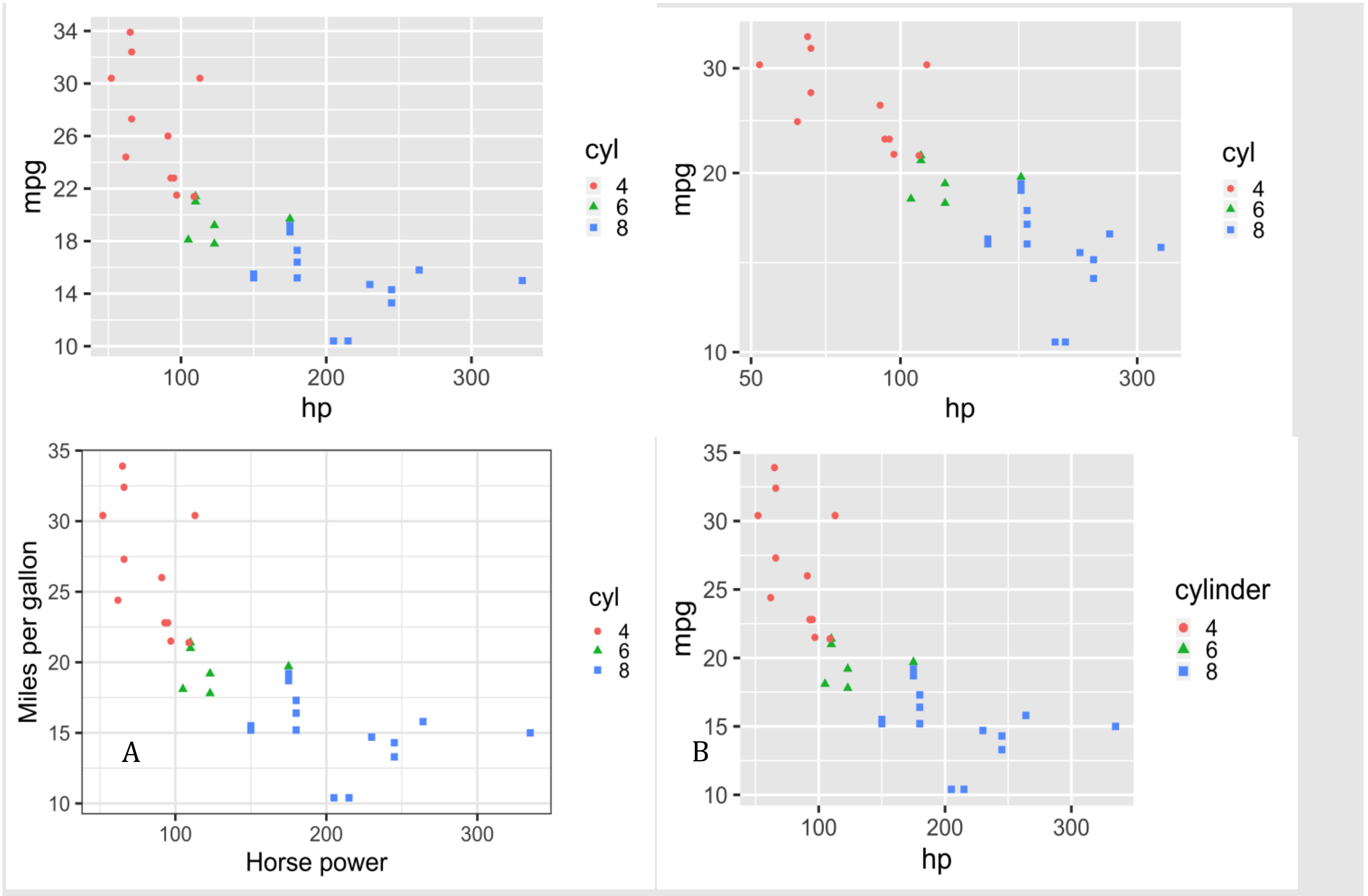


Figure 4. Example of Scatterplots Using the mtcars Data with Cosmetic Adjustments

Notes: (A) Specified breaks on the *y* axis, (B) log-scaled axes, (C) added axis labels and a black-and-white theme, and (D) enhanced legend keys.

The next set of figures provides examples of adding a data label layer (Figure 5) and examples of histograms and bar plots (Figure 6).

```
mtcars$car_model <- rownames(mtcars)

# add labels of car model for cars that have either hp > 200 or mpg > 25
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(aes(shape = cyl, color = cyl)) +
  ggrepel::geom_label_repel(aes(label = car_model),
                            data = filter(mtcars, hp > 200 | mpg > 25))

# example of boxplot
ggplot(mtcars, aes(x = am_char, y = wt)) +
  geom_boxplot() +
  geom_label_repel(aes(label = car_model),
                   data = filter(mtcars, wt > 4.5 | wt < 3, am == 0))
```

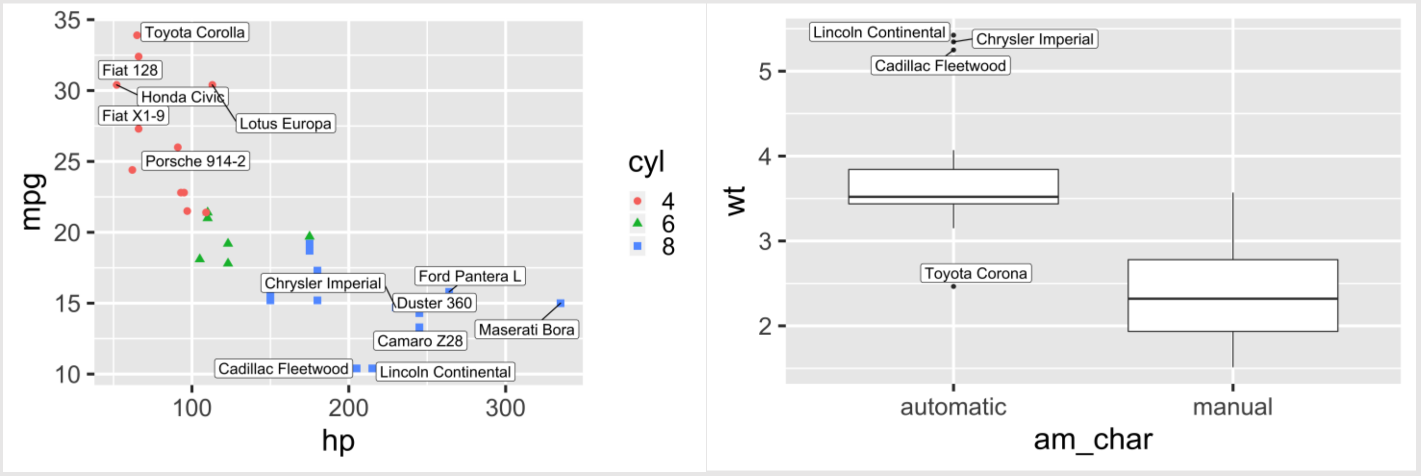


Figure 5. Example of Plots Using the mtcars Data with Selected Data-Point Labels

```
# examples of histograms
ggplot(mtcars, aes(x = wt, fill = am_char)) +
  geom_histogram(binwidth = .75)

ggplot(mtcars, aes(x = wt, color = am_char)) +
  geom_freqpoly(binwidth = .75, position="dodge", size = 2)

# examples of barplots
ggplot(mtcars, aes(x = cyl, fill = am_char)) + geom_bar()
ggplot(mtcars, aes(x = cyl, fill = am_char)) + geom_bar(position = "dodge")
ggplot(mtcars, aes(x = cyl, fill = am_char)) + geom_bar(position = "fill") + labs(y = "fract
ion")
```

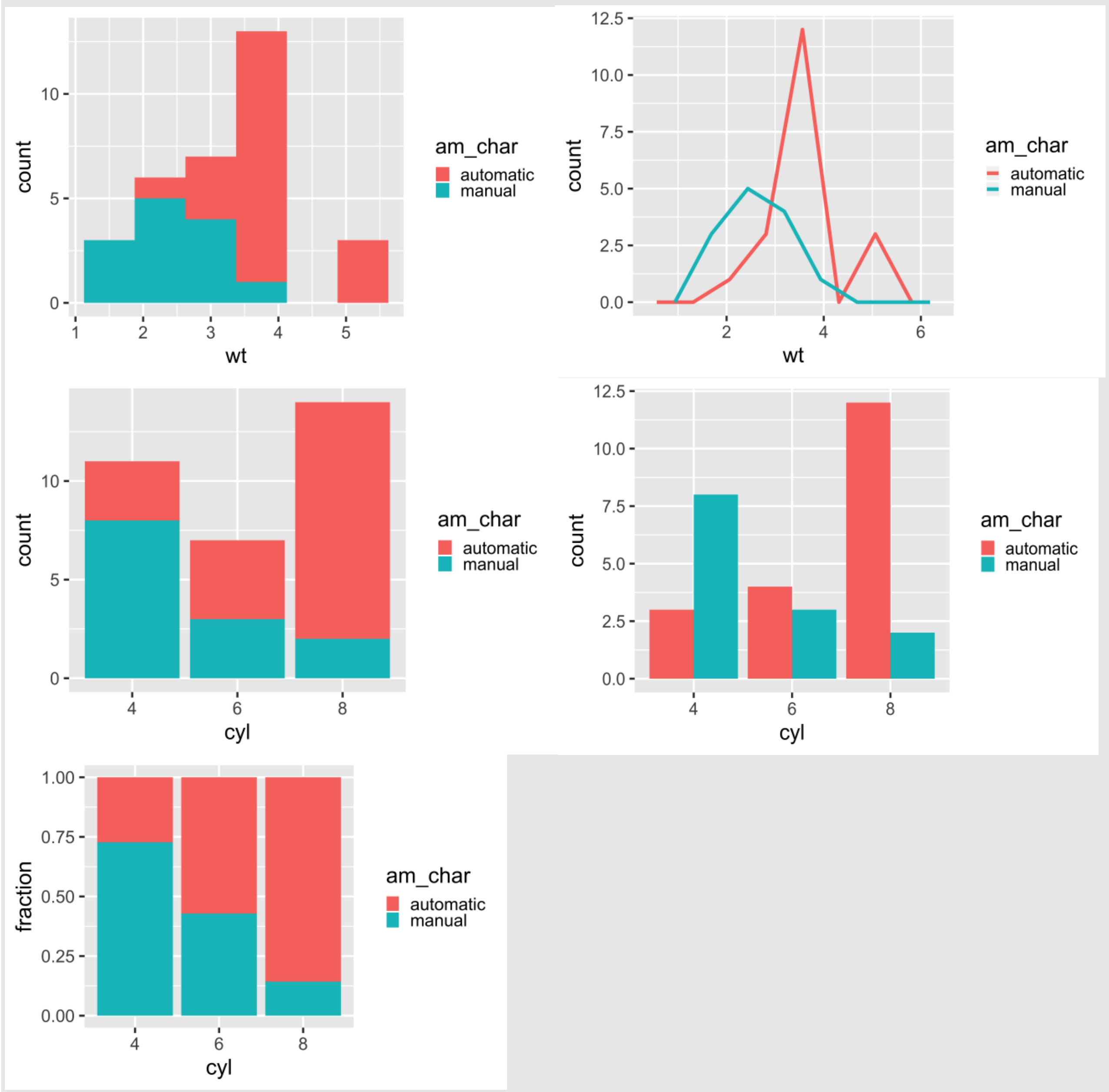


Figure 6. Example of Histograms (Classic Compound Bars and a Line Plot Style) and Bar Plots (Three Examples) Using the mtcars Data

Variables wt, cyl, and am_char refer to weight, the number of cylinders, and transmission type, respectively.

4 Data Exploration with *dplyr*

This section reviews essential functions for transforming data with **dplyr** and uses U.S. agriculture data for a demonstration of EDA that includes querying data, applying geospatial visualizations, and visual presentations of data summaries. Before we begin, let us note why exploring data is important and why tools of data transformation matter. Most statistical tools allow us to transform a data set by creating new variables, selecting specific subsets, sorting or grouping data, collapsing data into group-level statistics, or any sequential combination of those operations. And perhaps when combined with some data visualization, often by chance, the transformed data set may reveal new aspects of the data.

While curiosity-based exploration may seem like a luxury, it is necessary if we want to understand the data and discover the insights it provides. Only after a particular combination of data transformations, may certain aspects of the data be revealed or become noticeable. That should prompt subsequent questions like, "How do we know which data transformations to perform?" or "How can we tell whether we have uncovered all possible interesting aspects of the data?" A simple answer to both questions is, "We don't, but we should try our best." This is precisely why the tools of EDA matter. The easier and the simpler the tools are, the more frequently we use them and the more thoroughly we explore the data. The power of data visualization is multiplied by the ability and agility to transform the data at hand.

The tools of the *dplyr* package enable us to act nimbly, explore, and understand the data. That can make us feel like we are interacting with the data rather than merely transforming it. Before discussing why that may be the case, let us introduce the core R functions in the dplyr package:

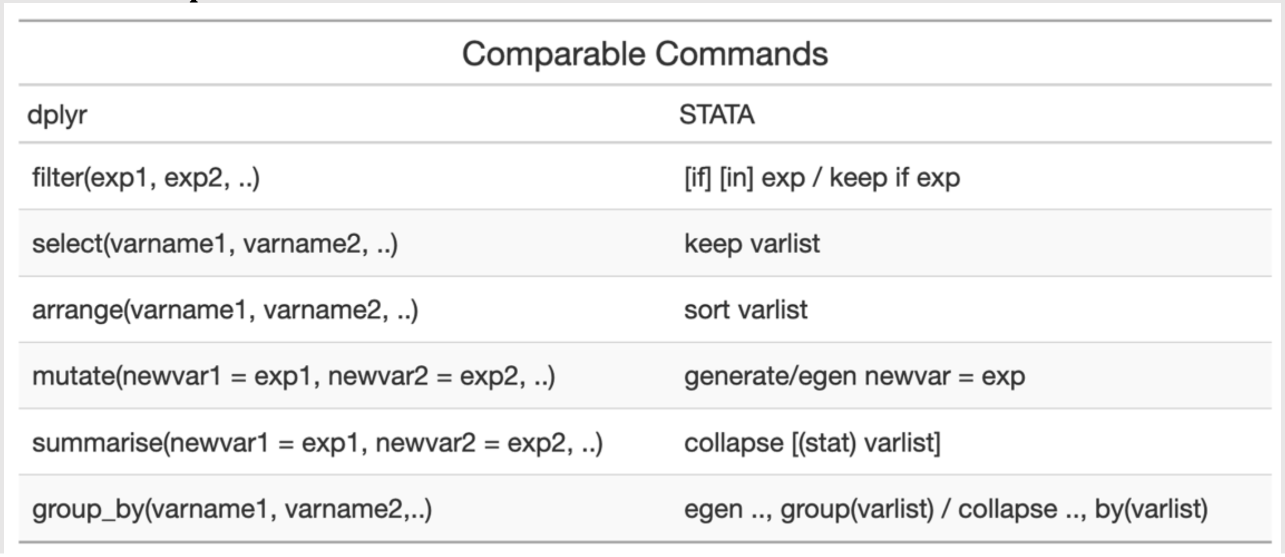
- *filter()*: extracts rows (observations) by logical vectors.
- *select()*: extracts columns (variables) by column names.
- *group_by()*: assigns rows into groups by column names.
- *mutate()*: creates new variables in a data frame.
- *summarise()*: collapses a data frame into summary statistics.
- *arrange()*: sorts row ordering based on column names.

These function names are self-descriptive: filter() makes a subset of the data set by extracting rows that meet specified conditions; select() extracts selected variables; group_by() creates a grouped data frame, which enables subsequent computations in mutate() and summarise() to be performed within each group; mutate() creates new variables through direct arithmetic operations of existing variables, canned functions, and user-defined functions; summarise() transforms a data set into statistics through canned functions or user-defined functions; and arrange() sorts the row order of the data set. These functions can be combined in any order to accomplish a desired data transformation. For example, one can extract a subset of the data by filter(), set groups by group_by(), compute summary statistics by summarise(), and use arrange() to sort the results.

Table 1 provides a comparison of these functions with the corresponding commands in Stata. Most applied economists would be very familiar with these data transformations, which is a helpful set of tools for getting started with *dplyr*. Here, we offer three reasons for why these *dplyr* functions can be perceived as more powerful than the corresponding functions in other programs such as Stata.

First, the *dplyr* functions are designed to be sequentially combined via *a pipe operator* (*%>%*), which makes the sequencing very smooth and natural to code. Each of the functions above takes a data frame object in the first argument and returns a data frame object, and this allows for piping, that is, applying functions sequentially by passing the output of one function into the first argument of the next. For

Table 1. Comparable Data Transformation Commands between R and Stata



Comparable Commands

dplyr	STATA
filter(exp1, exp2, ..)	[if] [in] exp / keep if exp
select(varname1, varname2, ..)	keep varlist
arrange(varname1, varname2, ..)	sort varlist
mutate(newvar1 = exp1, newvar2 = exp2, ..)	generate/egen newvar = exp
summarise(newvar1 = exp1, newvar2 = exp2, ..)	collapse [(stat) varlist]
group_by(varname1, varname2,..)	egen .., group(varlist) / collapse .., by(varlist)

example, *func3(func2(func1(data,...), ...), ...)* can be rewritten as *data %>% func1(...) %>% func2(...) %>% func3(...)*. Piping makes R code more readable and breaks down a complex data manipulation into a sequence of simple steps. Notably, we can read a sequence of operations in plain English by substituting the *%>%* symbol with *then*. For example, start with the data, then apply *func1(...)*, then *func2(...)*, and then *func3(..)*. This makes data exploration approachable (the user has an intuitive framework for coding the first few functions), expandable (functions are easy to add on), and even rewarding (the resulting code can accomplish complex data transformations).

Second, the simplicity in needing to remember just six functions is empowering for the user. These functions condense the essence of data transformations needed for exploring data. Remembering these functions and piping them allows us to perform a myriad of data transformations without dedicating much brain power to formulating the coding instructions.

Third, R's data management environment is conducive to performing a series of data transformation and visualization tasks without any commitment to altering the working copy of the data set. R separately handles the task of transforming data from the task of saving the transformed data under a given name. Piping allows us to execute a series of data tasks without needing to overwrite the working data set. When it is desirable to save transformed data (e.g., creating different data summaries or using them in subsequent calculations), it is straightforward to keep multiple data sets in the working environment (i.e., just give new names to outputs).

With those six commands presented above, we can approach data exploration through iterative trials of data transformations and visualizations through extracting subsets, grouping, sorting, generating variables, and computing data summaries. Each iteration, sparked by an inquisitive hypothesis, offers the potential to reveal new aspects of the data. The interesting data patterns, correlations, anomalies, and outliers revealed in one inquiry can lead to another line of inquiry. By allowing improvisations through EDA, we create a sense of interaction with the data. After repeated use, these tools in R can give one an increased sense of confidence and control to explore the data at hand.

4.1 Farm Data

We now move to our demonstrations with real data. In the rest of the section, we examine the U.S. Census of Agriculture (2017),[10] for which various summary data are publicly available at the country, state, and

[10] Available at https://www.nass.usda.gov/Publications/AgCensus/2017/index.php and also in the supplementary appendix.

county levels. For convenience, the downloaded data set is separated into a national-level data set *us17*, state-level data set *state17*, and county-level data set *county17*. For *us17*, specifying some variables by *select()* and printing the first five rows yields:

```
us17 %>% select(census_table, Sector, Commodity, Item, geog_level, Value) %>% print(n=5)
## # A tibble: 82,025 x 6
##   census_table Sector Commodity  Item                   geog_level  Value
##          <dbl> <chr>  <chr>      <chr>                  <chr>       <dbl>
## 1            1 ECONOM… FARM OPERA… FARM OPERATIONS - NUM… NATIONAL   2.04e6
## 2            1 ECONOM… FARM OPERA… FARM OPERATIONS - ACR… NATIONAL   9.00e8
## 3            1 ECONOM… FARM OPERA… FARM OPERATIONS - ARE… NATIONAL   4.41e2
## 4            1 ECONOM… AG LAND     AG LAND, INCL BUILDIN… NATIONAL   1.31e6
## 5            1 ECONOM… AG LAND     AG LAND, INCL BUILDIN… NATIONAL   2.98e3
## # … with 8.202e+04 more rows
```

Note that the national level data set alone contains over 80,000 rows. The state or county level data set will contain far more rows of data. To identify a variable of interest in a large data set like this, it is essential to have some understanding of its data structure. Two useful approaches here are to (1) become familiar with Quick Stats 2.0,[11] with which these data sets are consistently organized and (2) scan through published census of agriculture tables for its contents and organization.

Suppose that we are interested in the prevalence of small (those farms with less than $100,000 of sales) and nonsmall farms (for the sake of discussion, say, farms with greater than $100,000). The information needed for this is found in Table 2 of the U.S. and state census tables. We can extract the relevant information by specifying the table number in *filter()* and inspecting unique entries in the *Item* column:

```
# find the relevant Item
us17 %>% filter(census_table == 2) %>%
  select(Item) %>% unique()
## # A tibble: 144 x 1
##    Item
##    <chr>
##  1 COMMODITY TOTALS - OPERATIONS WITH SALES
##  2 COMMODITY TOTALS - SALES, MEASURED IN PCT OF FARM OPERATIONS
##  3 COMMODITY TOTALS - SALES, MEASURED IN $
##  4 COMMODITY TOTALS - SALES, MEASURED IN PCT OF FARM SALES
##  5 COMMODITY TOTALS - SALES, MEASURED IN $ / OPERATION
##  6 CROP TOTALS - OPERATIONS WITH SALES
##  7 CROP TOTALS - SALES, MEASURED IN PCT OF FARM OPERATIONS
##  8 CROP TOTALS - SALES, MEASURED IN $
##  9 CROP TOTALS - SALES, MEASURED IN PCT OF FARM SALES
## 10 GRAIN - OPERATIONS WITH SALES
## # … with 134 more rows
```

The information we need is a cross tabulation between the *Item* being "COMMODITY TOTALS—OPERATIONS WITH SALES" and the *Class*, two variables that contain the number of farms and the information about farm sales class. We use *filter()* to pinpoint the data we are seeking.

[11] Accessible at https://quickstats.nass.usda.gov/.

```
# find the relevant Item and Class
farm_class_US <- us17 %>%
    filter(
      census_table == 2,
      grepl("COMMODITY TOTALS - OPERATIONS WITH SALES", Item),
      !is.na(Class)
    ) %>% select(Class, Value)

farm_class_US
## # A tibble: 16 x 2
##    Class                                 Value
##    <chr>                                 <dbl>
##  1 FARM SALES: (LESS THAN 1,000 $)      603752
##  2 FARM SALES: (1,000 TO 2,499 $)       187949
##  3 FARM SALES: (2,500 TO 4,999 $)       185341
##  4 FARM SALES: (5,000 TO 9,999 $)       208074
##  5 FARM SALES: (10,000 TO 19,999 $)     174780
##  6 FARM SALES: (20,000 TO 24,999 $)      53438
##  7 FARM SALES: (25,000 TO 39,999 $)     100490
##  8 FARM SALES: (40,000 TO 49,999 $)      43623
##  9 FARM SALES: (50,000 TO 99,999 $)     119434
## 10 FARM SALES: (100,000 TO 249,999 $)   130932
## 11 FARM SALES: (250,000 TO 499,999 $)    87839
## 12 FARM SALES: (500,000 TO 999,999 $)    69703
## 13 FARM SALES: (1,000,000 OR MORE $)     76865
## 14 FARM SALES: (1,000,000 TO 2,499,999 $)  53611
## 15 FARM SALES: (2,500,000 TO 4,999,999 $)  14366
## 16 FARM SALES: (5,000,000 OR MORE $)      8888
```

Note that the national data set provides the aggregate record for the sales class of $5,000,000 or more as the most detailed information on larger farms. If similar operations are applied to the state or county level data, one would find that all sales classes above $1,000,000 and above $500,000 are aggregated, respectively.

Let's turn to county-level data. By continuing on the previous example, suppose that we want to count farms by a binary sales-class consisting of small farms (label *S*) versus not-small farms (label *NS*) at the county level. We do this by selecting relevant data, creating a new class variable (by comparing the sales class in the data to user-defined reference *class_S* that contains a vector of class names for those under $100,000 in sales), and summarizing the number of farms by county and the binary sales-class:

```
farms <- county17 %>%
  filter(
    census_table == 2,
    grepl("COMMODITY TOTALS - OPERATIONS WITH SALES", Item),
    !is.na(Class), Co_name != "NULL"
    ) %>%

  # create a new variable indicating sales < $100k
  mutate(class_S_NS = ifelse(Class %in% class_S, "S", "NS")) %>%
  group_by(St_code, St_name, Co_code, Co_name, class_S_NS) %>%
  summarise(Value = sum(Value, na.rm = T))

# show the top 10 county for the numbers of small farms
farms %>% filter(class_S_NS=="S") %>% arrange(desc(Value)) %>% head(n = 10)
## # A tibble: 10 x 6
## # Groups:   St_code, St_name, Co_code, Co_name [10]
##    St_code St_name Co_code Co_name   class_S_NS Value
##    <chr>   <chr>   <chr>   <chr>     <chr>      <dbl>
##  1 04      AZ      001     APACHE    S           5529
##  2 06      CA      073     SAN DIEGO S           4571
##  3 48      TX      367     PARKER    S           4521
##  4 04      AZ      017     NAVAJO    S           4181
##  5 48      TX      231     HUNT      S           4040
##  6 41      OR      005     CLACKAMAS S           4013
##  7 15      HI      001     HAWAII    S           3929
##  8 12      FL      083     MARION    S           3776
##  9 48      TX      497     WISE      S           3610
## 10 08      CO      123     WELD      S           3407

# show the top 10 county for the numbers of non-small farms
farms %>% filter(class_S_NS == "NS") %>% arrange(desc(Value)) %>% head(n = 10)
## # A tibble: 10 x 6
## # Groups:   St_code, St_name, Co_code, Co_name [10]
##    St_code St_name Co_code Co_name     class_S_NS Value
##    <chr>   <chr>   <chr>   <chr>       <chr>      <dbl>
##  1 42      PA      071     LANCASTER   NS          2382
##  2 06      CA      019     FRESNO      NS          2240
##  3 06      CA      107     TULARE      NS          1800
##  4 06      CA      077     SAN JOAQUIN NS          1414
##  5 06      CA      099     STANISLAUS  NS          1305
##  6 06      CA      047     MERCED      NS          1100
##  7 27      MN      145     STEARNS     NS          1091
##  8 19      IA      167     SIOUX       NS          1070
##  9 06      CA      097     SONOMA      NS           849
## 10 55      WI      043     GRANT       NS           828
```

When we compare where small (S) and nonsmall farms (NS) are numerous, the two lists of top counties are not geographically overlapping for these two farm classes. Summing up the number of farms within each binary sales class yields:

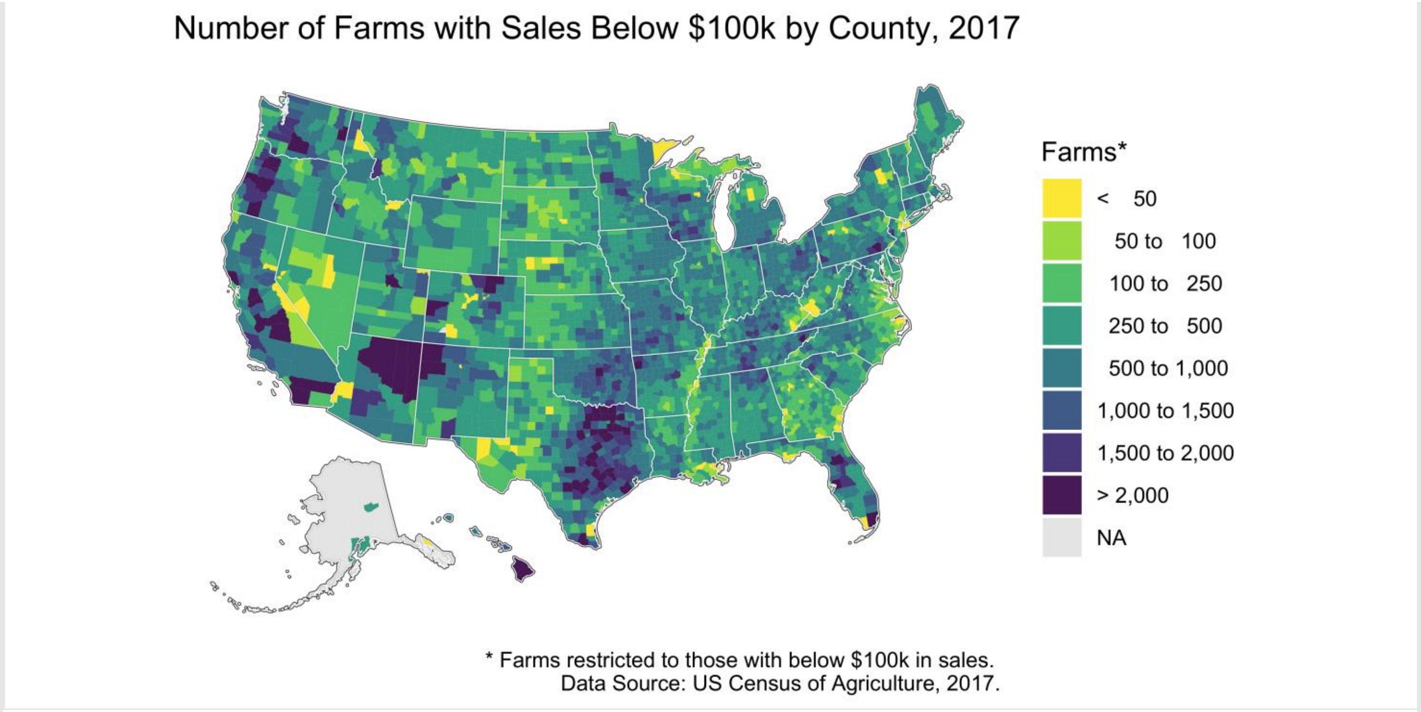
```
# total number of farms by class
farms %>% group_by(class_S_NS) %>%
  summarise(subtotal = sum(Value, na.rm = T)) %>%
  ungroup() %>%
  mutate(total = sum(subtotal, na.rm = T),
         fraction = round(subtotal / total, 2))
## # A tibble: 2 x 4
##   class_S_NS subtotal   total fraction
##   <chr>         <dbl>   <dbl>    <dbl>
## 1 NS           365339 2042220     0.18
## 2 S           1676881 2042220     0.82
```

Of the 2 million farms for which the census gathered data, roughly 1.68 million farms (82 percent) had less than $100,000 in revenues. The USDA defines a farm to be "any place from which $1,000 or more of agricultural products were produced and sold, or normally would have been sold, during the census year" (O'Donoghue et al. 2009). In fact, over 600,000 farms do not have sales above $1,000 in 2017, as shown in the first summary *farm_class_US* above. Although the definition of farms in USDA statistics has been debated previously, no change has been made (O'Donoghue et al. 2009).

One strength of R for agricultural data analysis is to be able to produce geographical representations of data. With county-level data paired with the state-county Federal Information Processing Standards (FIPS) codes, it is straightforward to project the data on maps. For instance, the following sample code shows how variable *var1* in data set *data* can be mapped at the county level:

```
# merge county level data with geographic data and generate a color-coded map
left_join(geo_county, data,  by = c("GEOID" = "FIPS")) %>%
    ggplot() +
    geom_sf(aes(fill = var1)) +
    coord_sf(datum = NA) + theme_minimal()
```

Here, *geo_county* contains the geometry data of U.S. county boundaries (which can be replicated by downloading any county-level information of the American Community Survey with *tidycensus* package). Layer *geom_sf()* handles the geometry aesthetic and here supplies a layer that fills county shapes with different colors depending on the value of *var1*. Additional layers *coord_sf(datum = NA)* and *theme_minimal()* instruct how to remove data plot graphics like axes and data plot area, giving a clean finish to the map output. Figures 7 and 8 provide examples of mapping the farm distributions using the binary revenue-class variable defined above.


Number of Farms with Sales Below $100k by County, 2017
Farms*
< 50
50 to 100
100 to 250
250 to 500
500 to 1,000
1,000 to 1,500
1,500 to 2,000
> 2,000
NA
* Farms restricted to those with below $100k in sales.
Data Source: US Census of Agriculture, 2017.

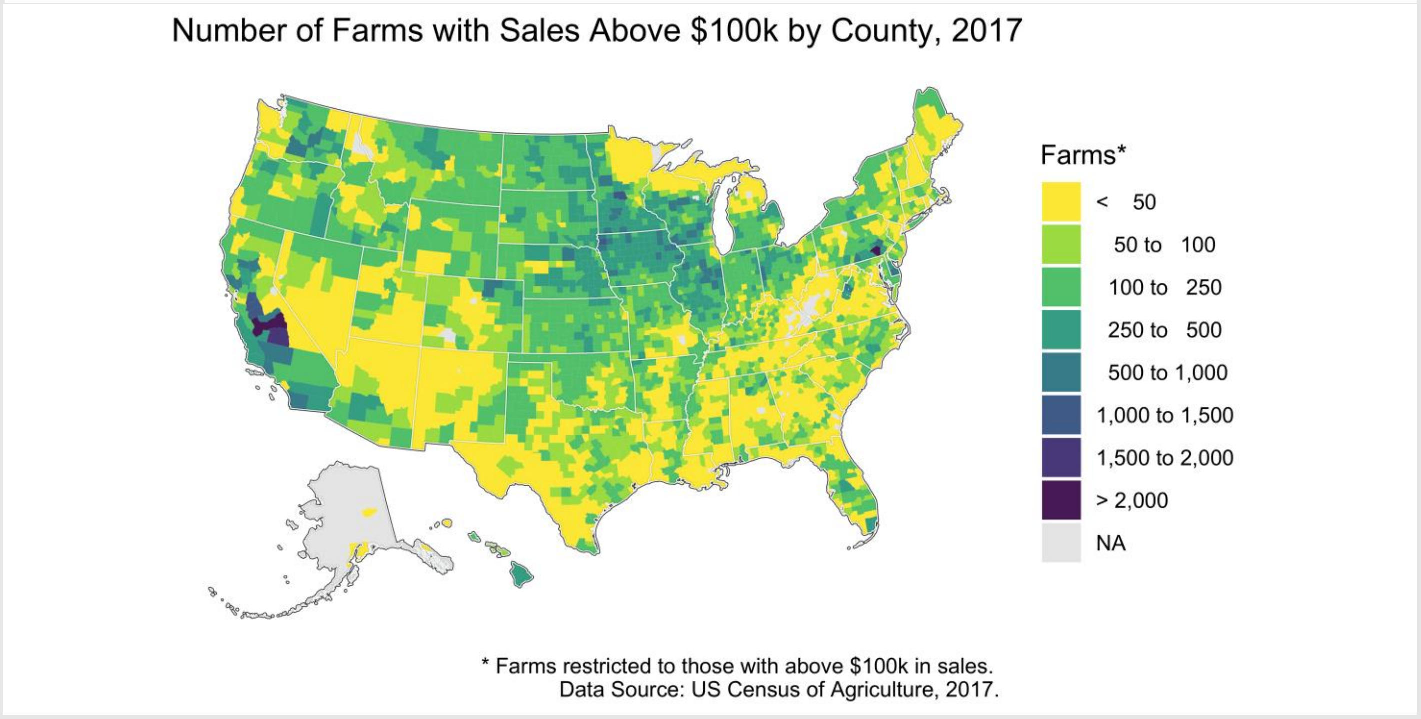
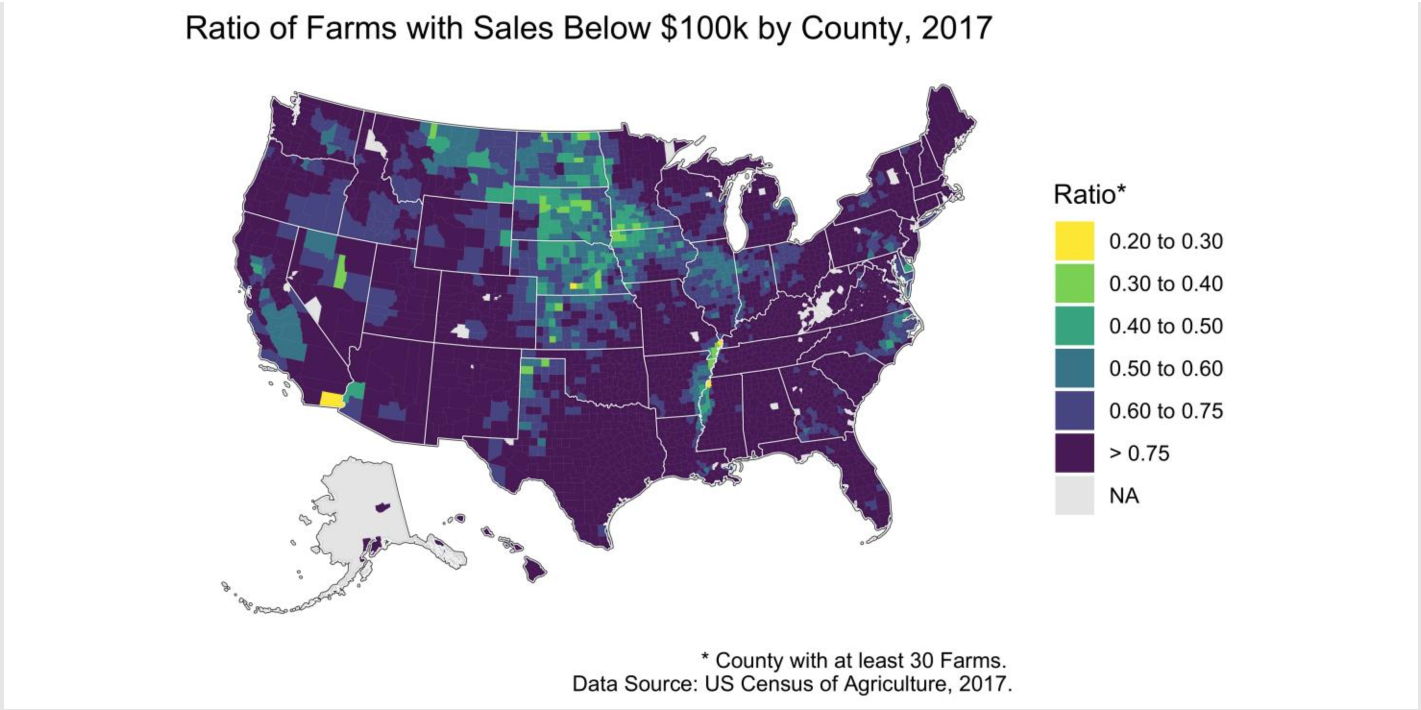

Number of Farms with Sales Above $100k by County, 2017
Farms*
< 50
50 to 100
100 to 250
250 to 500
500 to 1,000
1,000 to 1,500
1,500 to 2,000
> 2,000
NA
* Farms restricted to those with above $100k in sales.
Data Source: US Census of Agriculture, 2017.

Figure 7. Map of Farm Counts Using the Binary Sales-Revenue Class in the 2017 U.S. Census of Agriculture

The first map shows the distribution of farms with sales less than $100,000, and the second map shows the distribution of farms with sales above $100,000.

In addition to the raw farm counts, the next map considers the relative prevalence of the small and nonsmall farms (Figure 8). This approach may more clearly highlight the geographic concentrations of farms in different farm-size classes across counties, especially in terms of how the concept of a farm (i.e., the revenue size of active farming and what meets the criteria for being considered a farm in the U.S. Census of Agriculture database) systematically varies across geography.


Ratio of Farms with Sales Below $100k by County, 2017
Ratio*
0.20 to 0.30
0.30 to 0.40
0.40 to 0.50
0.50 to 0.60
0.60 to 0.75
> 0.75
NA
* County with at least 30 Farms.
Data Source: US Census of Agriculture, 2017.

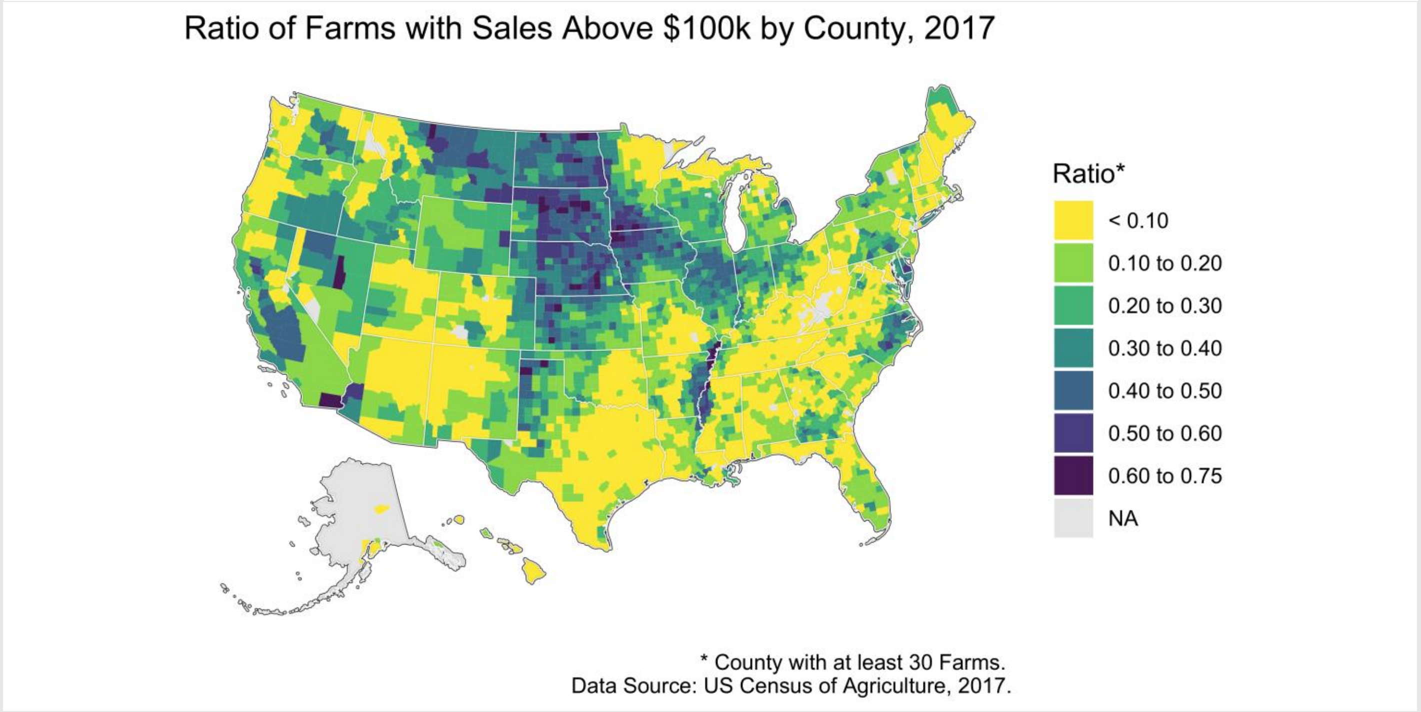

Ratio of Farms with Sales Above $100k by County, 2017
Ratio*
< 0.10
0.10 to 0.20
0.20 to 0.30
0.30 to 0.40
0.40 to 0.50
0.50 to 0.60
0.60 to 0.75
NA
* County with at least 30 Farms.
Data Source: US Census of Agriculture, 2017.

Figure 8. Map of Relative Farm Counts Using the Binary Sales-Revenue Classes in the 2017 U.S. Census of Agriculture

The two maps show the relative frequency of farms with sales below $100,000 (first), and the farms with sales above $100,000 (second).

Next we turn to differences across major farming industries. Suppose that we want to see how the concept of a farm differs across industries. We can examine the distributions of farm numbers and sales values this time by industry. In the first example, we show Sankey flow charts (we used the *flipPlots* package; Figures 9 and 10), which illustrate the contributions of different segments of data to the grand total like various streams combining into a river. Here, we add an intermediate layer that represents the subtotals by farm-sales class. For this purpose, we consider four levels of sales classes; marginal (less than

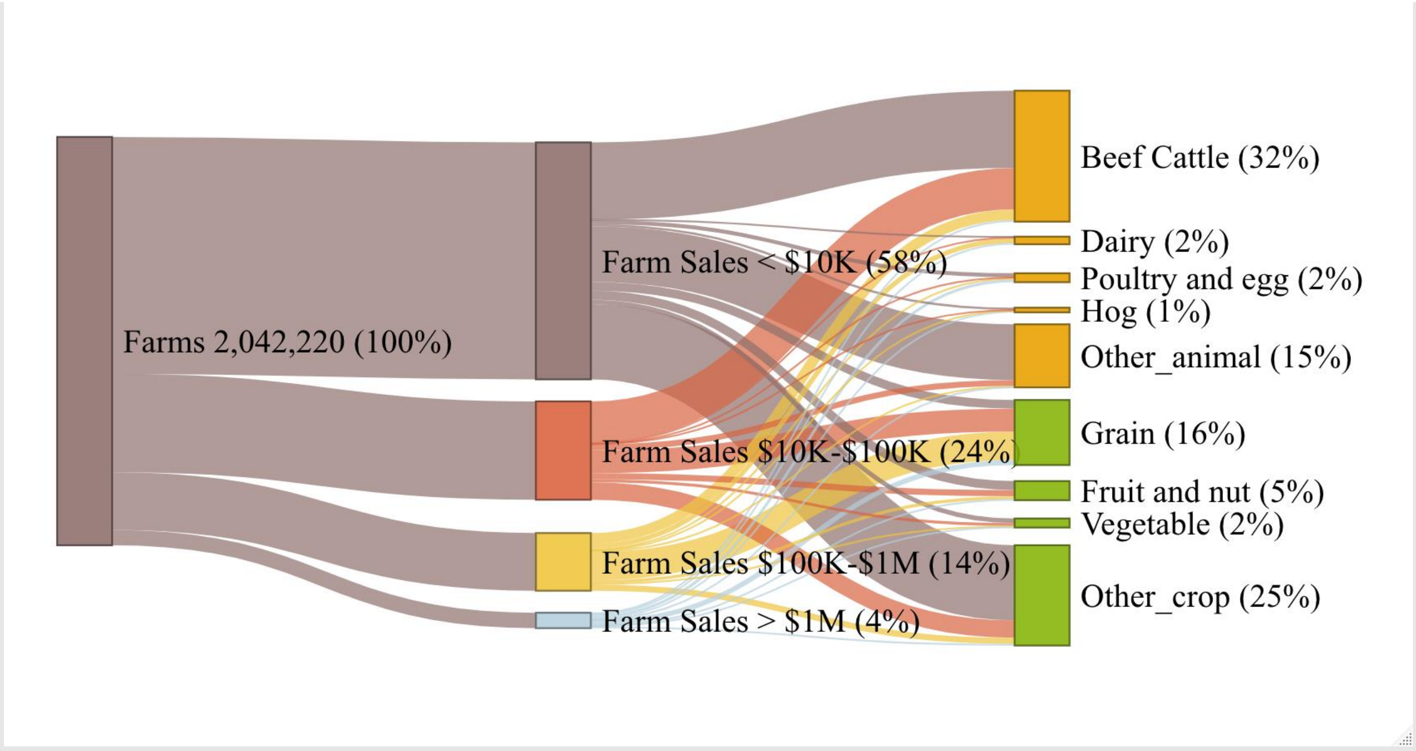


Figure 9. Sanky Flow Chart of Farm Counts by Sales and Industry from the 2017 U.S. Agricultural Census Data

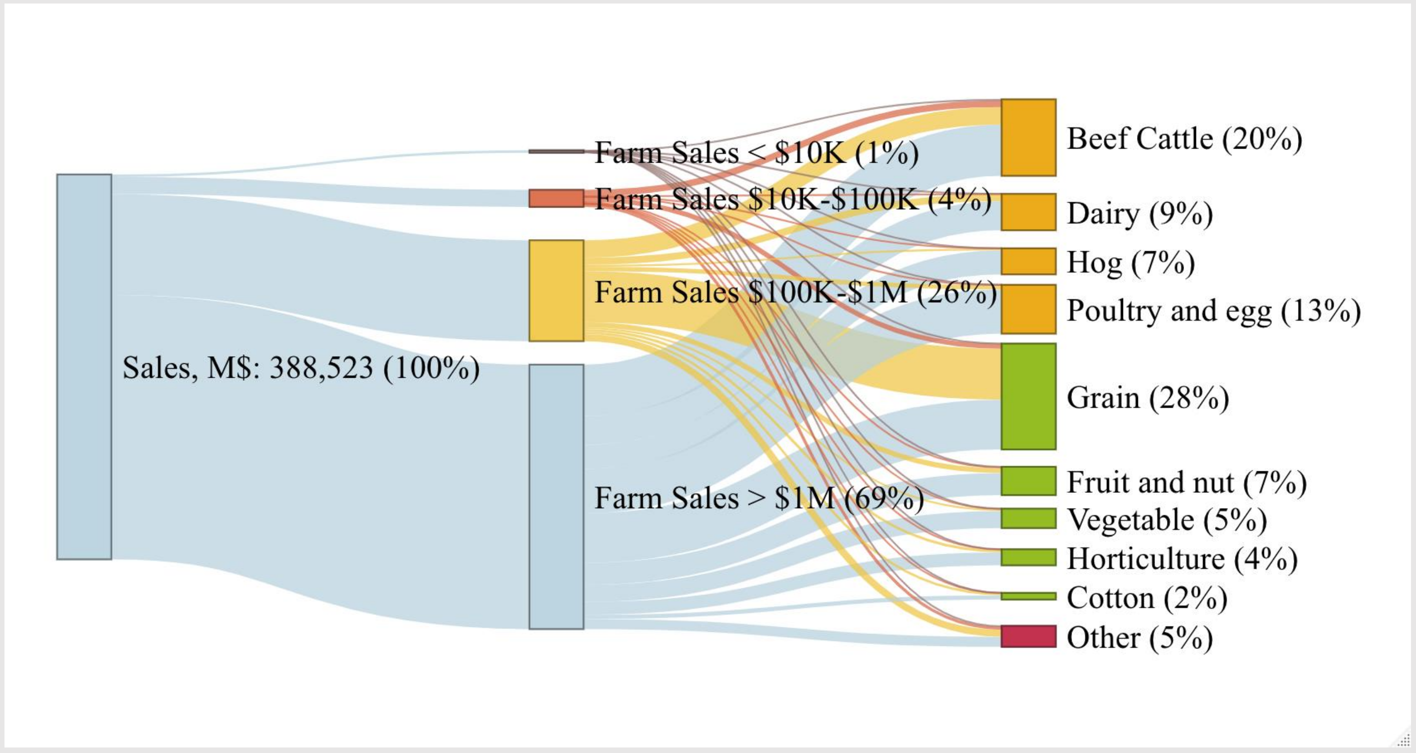


Figure 10. Sanky Flow Chart of Sales Values by Farm Sale Class and Industry from the 2017 U.S. Agricultural Census Data

$10,000), small ($10,000 to $100,000), medium ($100,000 to $1,000,000), and large (greater than $1,000,000). These charts show relationships among the farm numbers and sales values through the lens of farm size and by the industry.

Figure 9 shows that nearly 60 percent of the farms in the census are marginal producers with less than $10,000 in sales. Anyone who uses statistical information in the agricultural census must be aware of how the presence of these marginal farms impacts statistics like the averages per farm. On the other hand, the large farms with over $1,000,000 in sales revenues accounted for roughly 4 percent of the farm population, but produced nearly 70 percent ($268,000,000,000) of agricultural products in sales values (Figure 9 and 10). About 88 percent of the farms are classified as producers of grain, beef cattle, “other crop,” or “other animal” products (suggesting that only a small fraction of farms produce poultry and eggs, hogs, dairy, fruit and nuts, and vegetables). The majority of the medium-sized farms are grain producers. Grain production is unique in that its sales are not dominated by large-sized farms, as its total sales contribution is roughly equally split between medium- and large-sized farms.

4.2 How Does Farming Differ across States and Industries?

We next explore the characteristics of farm economies using industry statistics across states. It is common to see a ranking of states by sale values for a given industry. Here, we consider a slightly different comparison in which we visualize the relative size of a state’s crop and livestock sectors. By selecting certain variables from the state-level census data, we constructed the data set *df_NAICS* as organized by state and North American Industry Classification System (NAICS) code. In the following code example, we aggregate the sales revenue by state and NAICS category (i.e., crop or livestock), converting the data set into the “wide” format by distributing the sales value into “crop” and “livestock” variables, and then plot the data with the annotation of state names if the state exceeds certain sales value thresholds (Figure 11):

```r
# see "ag_examples.R" for creating data set "df_NAICS"
load(file="data sets/df_NAICS.RData")

crop_vs_animal <-
  df_NAICS %>% filter(!is.na(NAICS_cat)) %>%
  group_by(St_code, St_name, USDA_region, NAICS_cat) %>%
  summarise(revenue_sales = sum(revenue_sales, na.rm = T) / 10^9) %>%
  pivot_wider(names_from = NAICS_cat, values_from = revenue_sales)

crop_vs_animal %>%
  ggplot(aes(x = Crop, y = Livestock, color = USDA_region, shape = USDA_region)) +
  geom_point() +
  geom_label_repel(aes(label = St_name), show.legend = FALSE,
                   data = crop_vs_animal %>% filter(Crop > 6 | Livestock > 7)) +
  labs(x = "Crop Agriculture Revenue, $ billion",
       y = "Livestock Agriculture Revenue, $ billion",
       caption = "Data Source: US Census of Agriculture, 2017.")
```

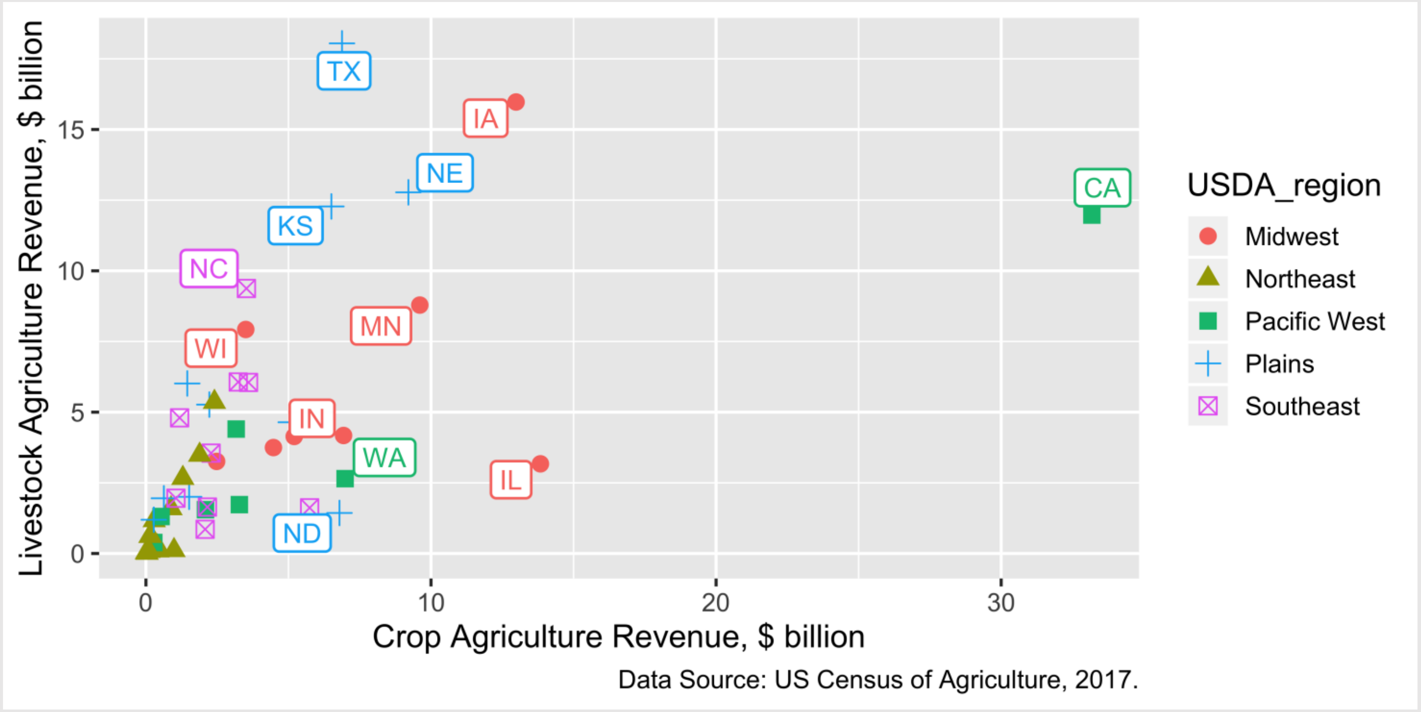


Figure 11. Livestock versus Crop Output by State (Selectively Labeled) from the 2017 U.S. Agricultural Census Data

It is clear that California is an exceptionally large agricultural state in both crop and livestock production. Also, one can see that Illinois, Washington, and North Dakota are specialized in crop production; Texas, Kansas, North Carolina, and Wisconsin are specialized in livestock production; and Iowa, Nebraska, and Minnesota are relatively balanced between the revenues from crop and animal agriculture (Figure 11).

In gathering various USDA National Agricultural Statistics Service (NASS) and census data, it is convenient to directly download them using an API (e.g., using the *rnassqs* package). The following is an example for obtaining the aggregate land asset value and net farm income for the poultry industry from the agricultural census data:

```
library(rnassqs)
NASSQS_TOKEN <- "C9B668A9-3062-..." # use your token
nassqs_auth(key = NASSQS_TOKEN)

# check asset and profitability of poultry sector
asset_profit_poultry <- nassqs(list(
  source_desc = "census",
  agg_level_desc = "national",
  domaincat_desc= "NAICS CLASSIFICATION: (1123)",
  short_desc = c("AG LAND, INCL BUILDINGS - ASSET VALUE, MEASURED IN $",
                 "INCOME, NET CASH FARM, OF OPERATIONS - NET INCOME, MEASURED IN $"),
  year = c(2012, 2017))) %>%
  select(sector_desc, short_desc, state_alpha, year, commodity_desc, Value)

# note: only 2012 and 2017 data are available
```

Next, suppose that we ask, “what does it take for a farm to thrive?” To explore this question, it is instructive to compare the average utilization of capital and labor per operator across states and agricultural industries. Here we define capital as the sum of the total asset value of land, buildings, and machinery for crop farming. For livestock farms, we add the value of livestock inventory for poultry (broiler chickens, nonbroiler chickens, and turkeys), hogs, dairy cows, and beef cattle using NASS survey and census statistics. For the poultry industry, we further add an estimated value of facility (for processing, hatchery, and feed mills that are largely owned by integrators) at the estimated rate of $3.5 per chicken-equivalent production (using the approximate rate based on the reporting by Wood 2018). Note that these asset values are only a crude approximation (Figures 12 and 13).

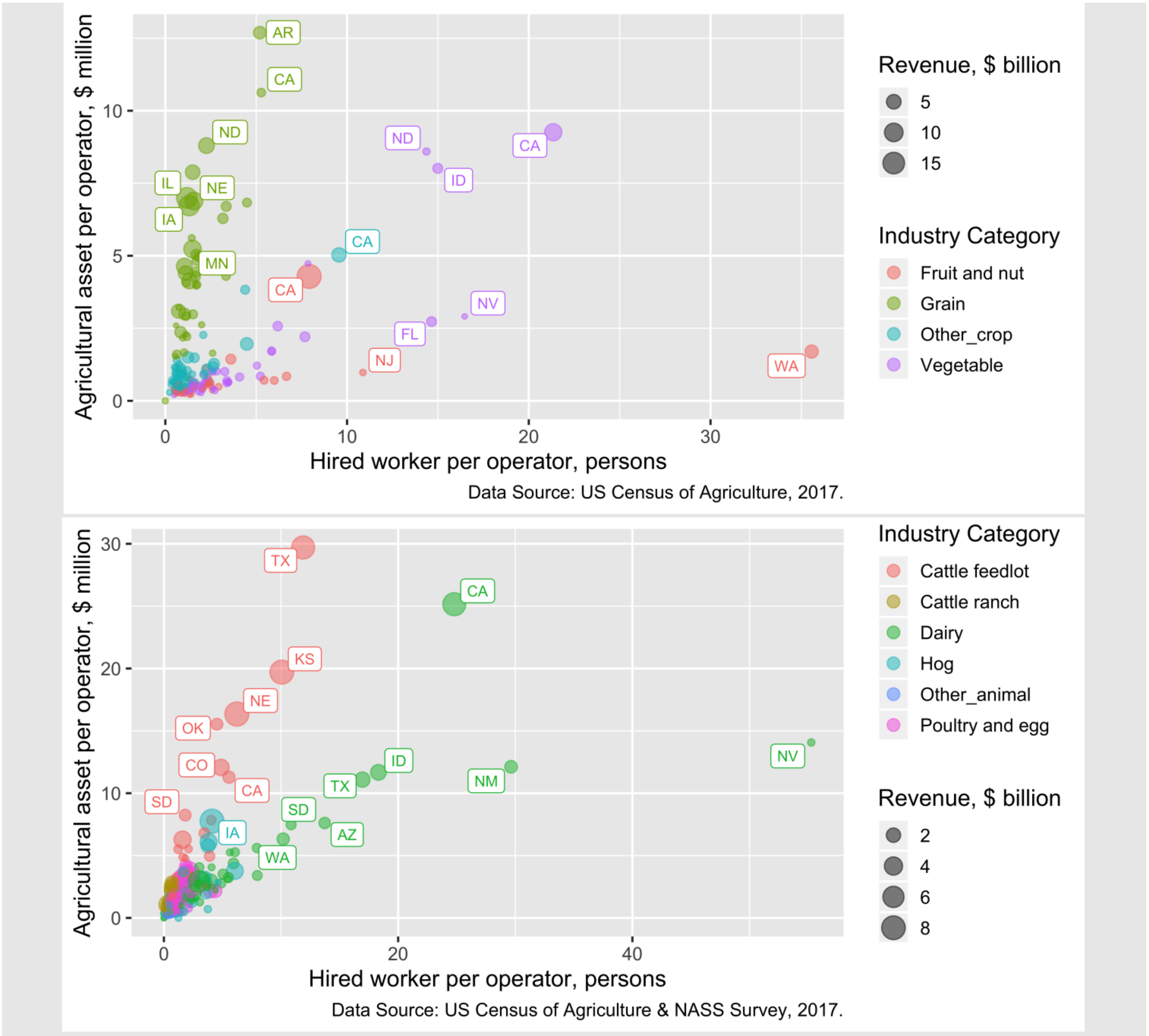


Figure 12. Capital and Labor per Operator by State and Agricultural Industry from the 2017 U.S. Agricultural Census and NASS Survey

Note: The top plot shows the data plot for crop industries, and the bottom plot shows that for livestock industries.

Grain production is more capital intensive than other types of crop farming, whereas fruit and nut production tends to be more labor intensive (Figure 12). In most states, grain producers are likely to require from $2,000,000 to 5,000,000 of capital asset, for which much of the value can be attributed to the value of the land. The data points for the "other crop" category are clustered together near zero except California, potentially because this category contains many marginal producers with less than $10,000 in sales.

For livestock agriculture, it is clear that cattle feedlot production is capital intensive, in which much of the capital is tied to the value of cattle inventory. In contrast, the data points for cattle ranch operations are clustered near zero. Indeed, beef producers are very different between ranch and feedlot operations since a typical feedlot manages much larger herds of cattle than a typical ranch. Dairy production is both capital and labor intensive; the average dairy operator in California, Nevada, New Mexico, Idaho, and Texas employs over $10,000,000 of assets and near 20 hired workers or more. In poultry and egg production, the notion of a farm operator itself is rather different because many producers operate under contracts with larger integrators such as Tyson, Pilgrim's Pride, and Perdue. According to Alonzo (2016), in 2015, the top five integrators had over 60 percent of market share in the poultry and egg industry.

In the plot above, we see that the data points for some types of operations like grain, dairy, and cattle feedlot production, visually line up with underlying linear trends. We can obtain an ordinary least squares (OLS) estimate of this trend using the *lm()* function for linear models.

```
# OLS estimation by lm(.) function

# Regress asset dollars on hired labor for dairy data
lm( formula = asset_per_unpaid ~ hired_to_unpaid,
    data  = df_NAICS_simple %>%
      filter(revenue_sales > .01,
             NAICS_simple == "Dairy")
  ) %>% summary()
##
## Call:
## lm(formula = asset_per_unpaid ~ hired_to_unpaid, data = df_NAICS_simple %>%
##     filter(revenue_sales > 0.01, NAICS_simple == "Dairy"))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3776 -0.9915 -0.4703  0.4672 14.1909
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1.62684    0.48002   3.389  0.00147 **
## hired_to_unpaid  0.37662    0.04181   9.009 1.23e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.695 on 45 degrees of freedom
## Multiple R-squared:  0.6433, Adjusted R-squared:  0.6354
## F-statistic: 81.16 on 1 and 45 DF,  p-value: 1.231e-11
```

lm() produces a linear model class object, on which applying the *summary()* function gives an informative output with a table of coefficients and common goodness-of-fit statistics. Here, we see that for each hired farm worker, the estimated slope coefficient implies that a typical dairy farm would employ $377,000 worth of capital asset per hired farm worker. Let's add a few more variables to this regression, such as regional fixed effects and a debt-to-income ratio:

```
# Add more variables: region dummies, debt-to-income ratio
lm( formula = asset_per_unpaid ~
      hired_to_unpaid + USDA_region + I(debt_at_5pct/revenue_sales),
    data  = df_NAICS_simple %>%
      filter(revenue_sales > .01,
             NAICS_simple == "Dairy")
  ) %>% summary()
##
## Call:
## lm(formula = asset_per_unpaid ~ hired_to_unpaid + USDA_region +
##     I(debt_at_5pct/revenue_sales), data = df_NAICS_simple %>%
##     filter(revenue_sales > 0.01, NAICS_simple == "Dairy"))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7925 -0.8585  0.0090  0.5823 12.8283
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    0.22568    1.91030   0.118   0.9065
## hired_to_unpaid                0.30129    0.05387   5.593 1.76e-06 ***
## USDA_regionNortheast           0.04172    1.22488   0.034   0.9730
## USDA_regionPacific West        3.59442    1.66197   2.163   0.0366 *
## USDA_regionPlains              1.48626    1.30884   1.136   0.2629
## USDA_regionSoutheast          -0.09535    1.40162  -0.068   0.9461
## I(debt_at_5pct/revenue_sales)  2.12396    2.50791   0.847   0.4021
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.617 on 40 degrees of freedom
## Multiple R-squared:  0.701,  Adjusted R-squared:  0.6562
## F-statistic: 15.63 on 6 and 40 DF,  p-value: 3.852e-09
```

lm() treats character-string variables as factor/categorical variables and inserts indicator dummies for each group. Also, to create a new variable from manipulating existing variables, one can use the *I(.)* operator in the linear model formula. The estimates show that after accounting for regional differences in the intercept and the relative use of debt to sales revenues, the average dairy farm capital asset is about $301,000 per hired worker.

Last, we briefly turn to the capital structure and return on asset in farming (Figure 13). Keep in mind that agricultural commodity prices vary from year to year, which causes the profitability to fluctuate. Some states had a particularly profitable year in vegetable, fruit, and nut production in 2017. The poultry and egg industry also had a particularly profitable year (note: the industry's net income doubled from 2012 to 2017, according to the Census of Agriculture). Dairy producers in states with large-sized dairy operations attained relatively high returns, while they were also highly leveraged (Figure 13).

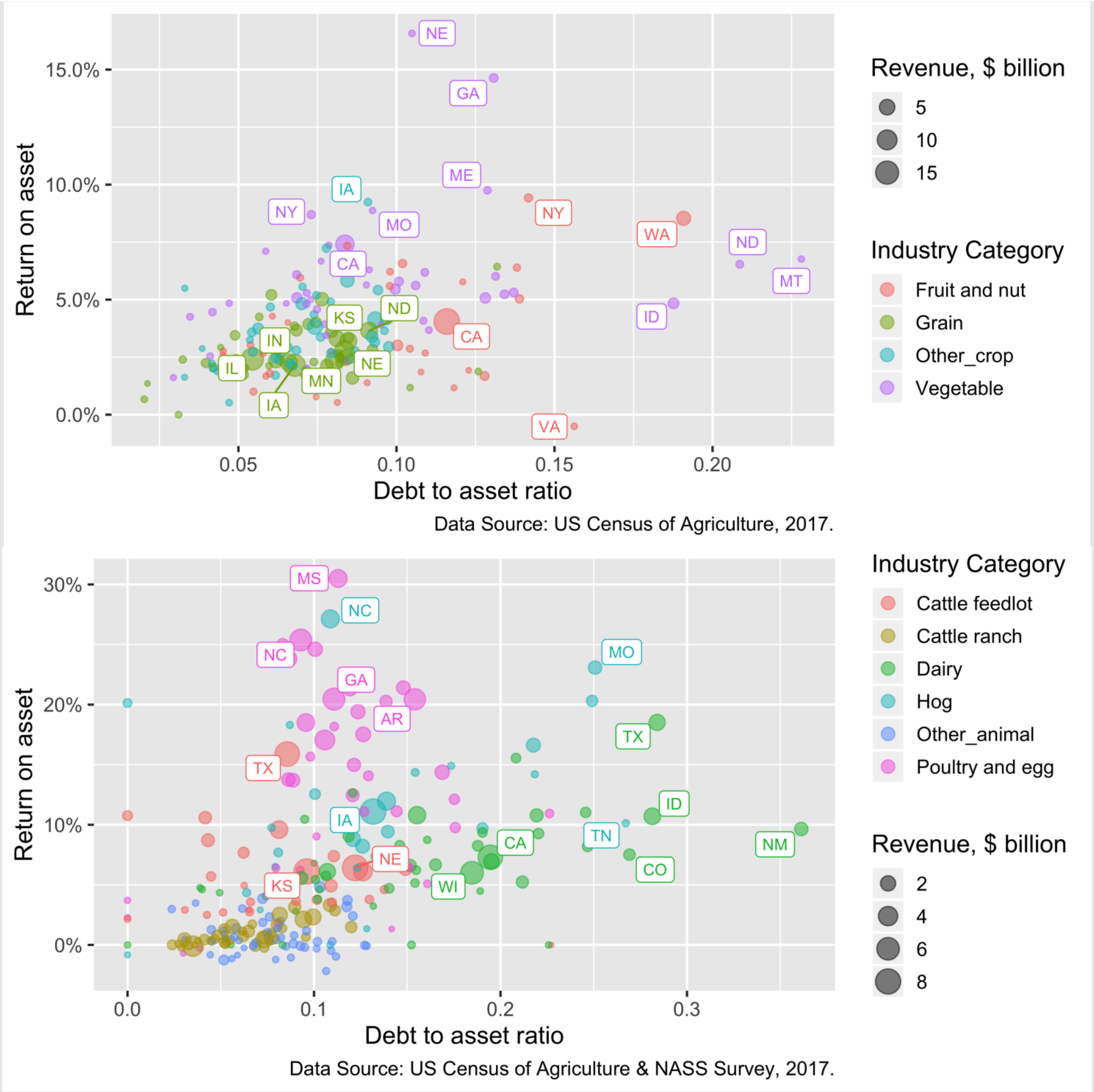


Figure 13. Return and Debt per Asset by State and Agricultural Industry from the 2017 U.S. Agricultural Census and NASS Survey

Note: The first plot shows the data plot for crop industries, and the second plot shows data for livestock industries.

5 Analytical Demonstration

For further illustration, this section presents an example of analytical data exploration on the topic of rural population change. In particular, we investigate whether there are systematic relationships between the intensification of grain farming and rural depopulation during the period 1972–2017. In preparation of the data set, we selected the data for 1972 as the beginning of this time span because the NASS survey data in 1970 had a large number of missing data points. For the data beyond 1982, we assigned missing grain production values with zero if the county had a nonmissing value in the 1982 data. All grain production values were expressed in 2017 dollars.

We first generate two maps: one for the grain production by county in 1972 and the other for the change in grain production from 1972 to 2017 (Figure 14). The first map also shows that much of the Midwest had highly active grain production in 1972. The second map highlights a relative decline in grain production in many parts of the country, while the Midwest and a part of the South increased their grain production.

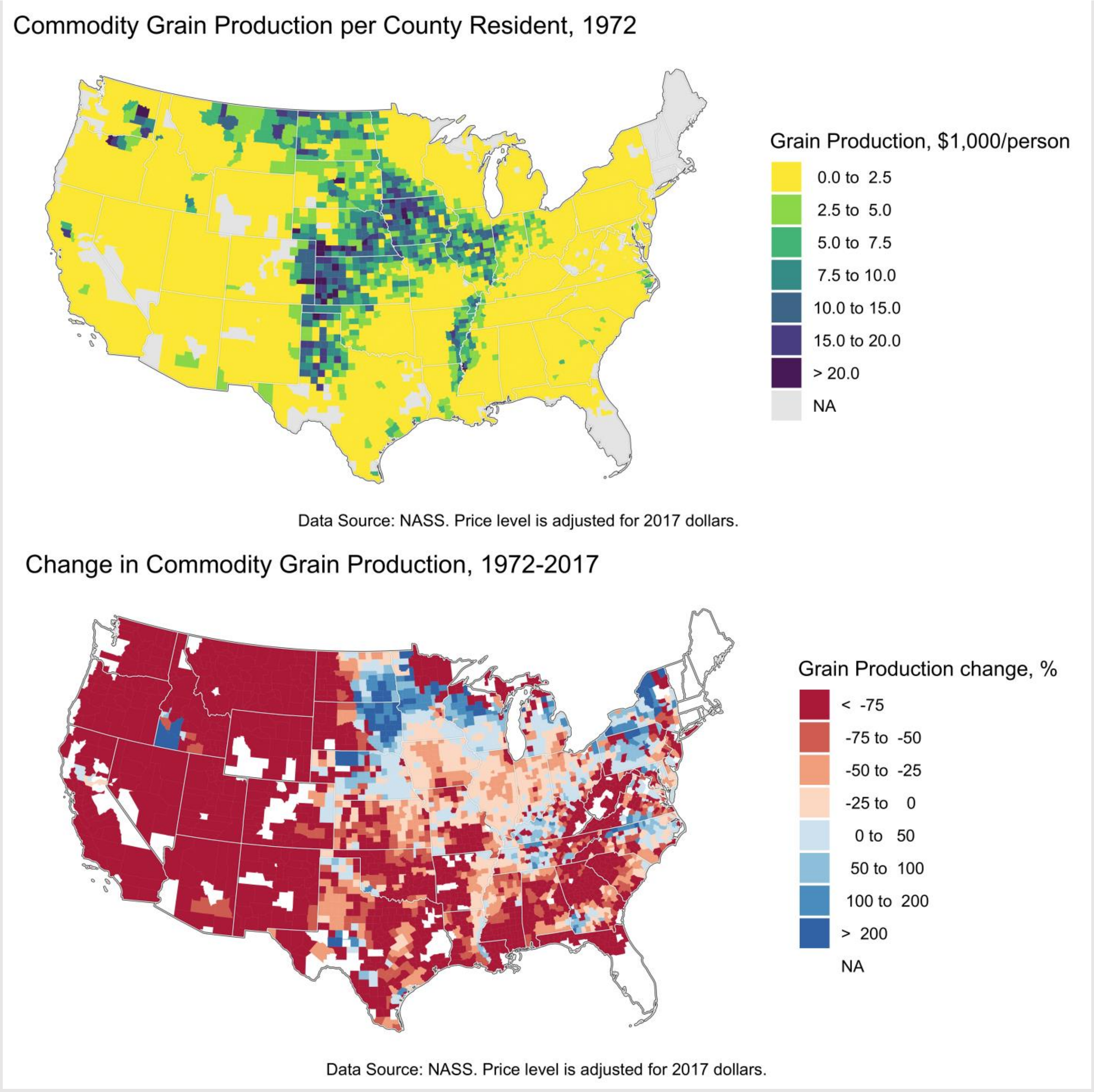

Commodity Grain Production per County Resident, 1972
Grain Production, $1,000/person
0.0 to 2.5
2.5 to 5.0
5.0 to 7.5
7.5 to 10.0
10.0 to 15.0
15.0 to 20.0
> 20.0
NA
Data Source: NASS. Price level is adjusted for 2017 dollars.
Change in Commodity Grain Production, 1972-2017
Grain Production change, %
< -75
-75 to -50
-50 to -25
-25 to 0
0 to 50
50 to 100
100 to 200
> 200
NA
Data Source: NASS. Price level is adjusted for 2017 dollars.

Figure 14. Map of Grain Production in 1972 and Production Change from 1972 to 2017

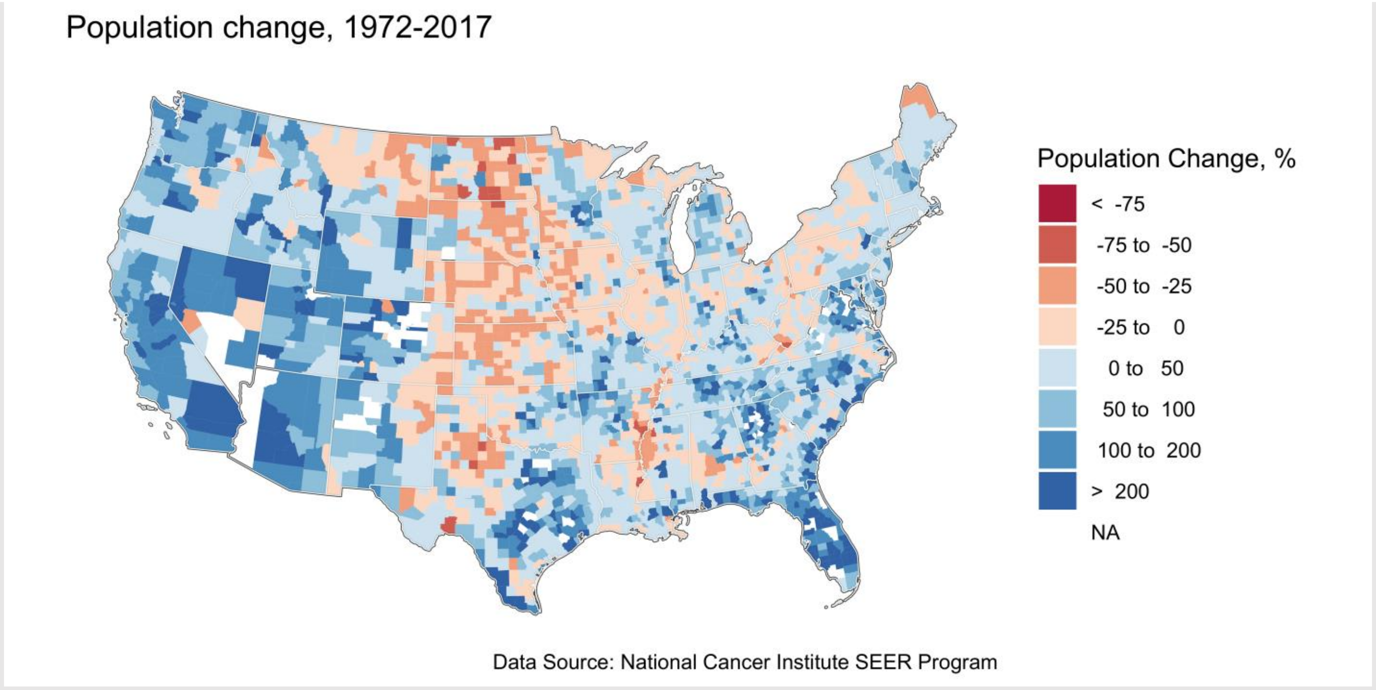


Figure 15. Map of Population Change, 1972–2017

We next map the overall population change during the same period (Figure 15). It is clear that the Midwest experienced the most significant population loss as a region. The two sets of maps together appear consistent with a narrative that increased mechanization of grain production required fewer and fewer laborers, which most severely affected the population in the Midwest (Johnson and Fuguitte 2000; Walzer 2003; White 2008; Longwoth 2009).

To further investigate the relationship between grain farming and population change, we plot county-level data against per capita grain production. In Figure 16, the top row contains a data plot of the raw data points (A) and a plot in the log-scale on the horizontal axis (B). The latter plot appears to suggest a negative correlation between the population change and the grain production per person in 1972. This correlation may be spurious because grain production per person may be affected by declining county population trends. Thus, we substitute this measure with the total grain production in the county (C) as well as the percentage change in grain production for 1972–2017 (D). For the latter, the cluster of data points at -100 percent change represents the counties that produced some grain in 1972 and had no sales records in 2017. These data plots seem to corroborate weak negative correlations between grain production and population change.

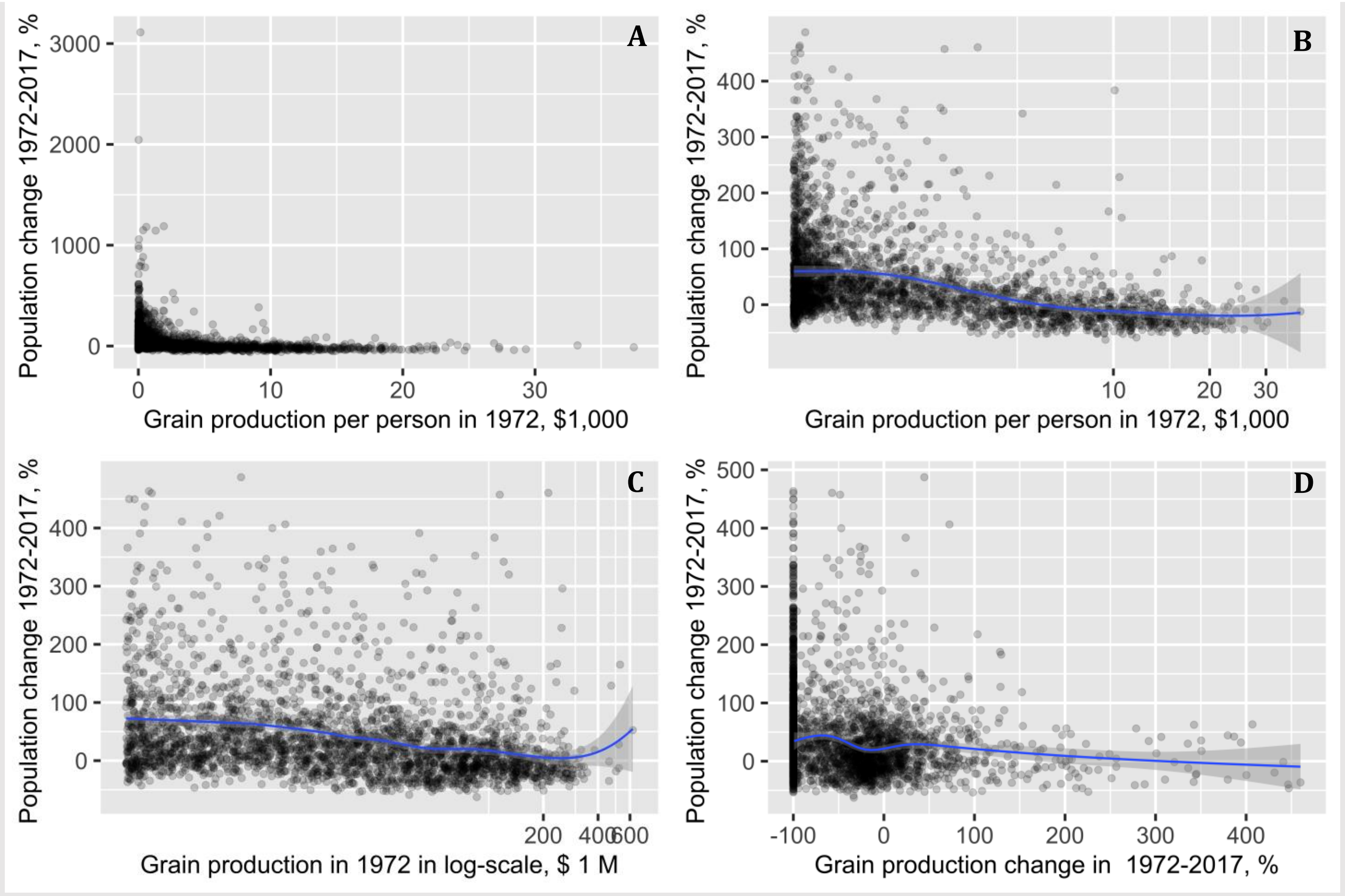


Figure 16. Scatter Data Plots of Grain Production and Population Change
Note: Top row figures use grain production per person in 1972 on the horizontal axis in the raw data scale (A) and the logarithmic scale (B). The bottom figures use grain production in log-scaled dollars (C) and grain production in percentage change (D).

Analytically, let us consider an ordinary least squares regression of the form

$$y_{is} = \alpha_s + \mathbf{x}_{is}\boldsymbol{\beta} + \varepsilon_{is}$$

where y_{is} is population change in county i in state s from 1972 to 2017, α_s are state fixed effects, $\mathbf{x}_{is}$ a vector of covariates, and ε_{is} an error term. For $\mathbf{x}_{is}$, we include grain production in 1972, the change in grain production from 1972 to 2017, and a dummy variable corresponding to the value of -100 percent changes. Given that some counties are much larger than others in terms of land area or in terms of population, we consider two models based on the county-level grain production per person (column (1)) along with total grain production (column (2)). We estimate the above equation using the linear regression model function *lm()* and summarize selected coefficients using the *stargazer* package.

```
lm_1 <- lm(pop_tot_ch_pct_72_17 ~ ln_grain_prod_person_1972 + grain_ch_pct_72_17 +
             (grain_ch_pct_72_17 == -100) + St_name,
           data = df_pop_grain)

lm_2 <- lm(pop_tot_ch_pct_72_17 ~   ln_grain_prod_1972 + grain_ch_pct_72_17 +
             (grain_ch_pct_72_17 == -100) + St_name,
            data = df_pop_grain)
```

Table 2. Estimate of Grain Production and Populations Change from 1972-2017

Variable	Population Change, % 1972–2017	
	Model 1	**Model 2**
Log of grain production per capita, 1972	-34.293***	
(ln_grain_prod_person_1972)	(2.101)	
Log of grain production, 1972		-7.140***
(ln_grain_prod_1972)		(1.282)
Change in grain production, 1972–2017	-0.096***	-0.090***
(grain_ch_pct_72_17)	(0.024)	(0.025)
Indicator for ceased grain production	-8.785**	-0.706
(grain_ch_pct_72_17==-100)	(4.049)	(4.684)
State fixed effects	Yes	Yes
Observations	2,727	2,727
Adjusted R squared	0.286	0.224
Residual Std. Error	65.347	68.124

Note: Statistical significance $^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$. The two models differ in the grain production variable specified either as per capita within the county or the county total.

The results suggest negative associations between the grain production variables and population change, while controlling for unobservable fixed factors at the state level (Table 2). In terms of magnitudes, the first model indicates that a 10 percent higher grain production *per person* in 1972 is associated with an additional 3.4 percent reduction in the county population, while the second model suggests a 10 percent higher grain production in the county total is similarly associated with a 0.7 percent reduction. Of the models, the first model is more closely aligned with the relative importance of grain production in the county's economy and is here shown to be more strongly negatively correlated with the population change. The two models also suggest that a 10 percent *increase* in grain production from 1972 to 2017 is associated with an additional 0.9 to 1.0 percent decline in the population.

To examine the geographic distribution of the errors, we add the estimation errors to the data set by the *add_residual()* function from the *modelr* package:

```r
# add model predictions, except states that have no grain production
df_pop_grain_res <- df_pop_grain %>%
  filter(!(St_name %in% c("CT", "DC", "MA", "ME", "NH", "RI", "VT"))) %>%
  add_residuals(lm_1, var = "resid_lm_1") %>%
  add_residuals(lm_2, var = "resid_lm_2")
```

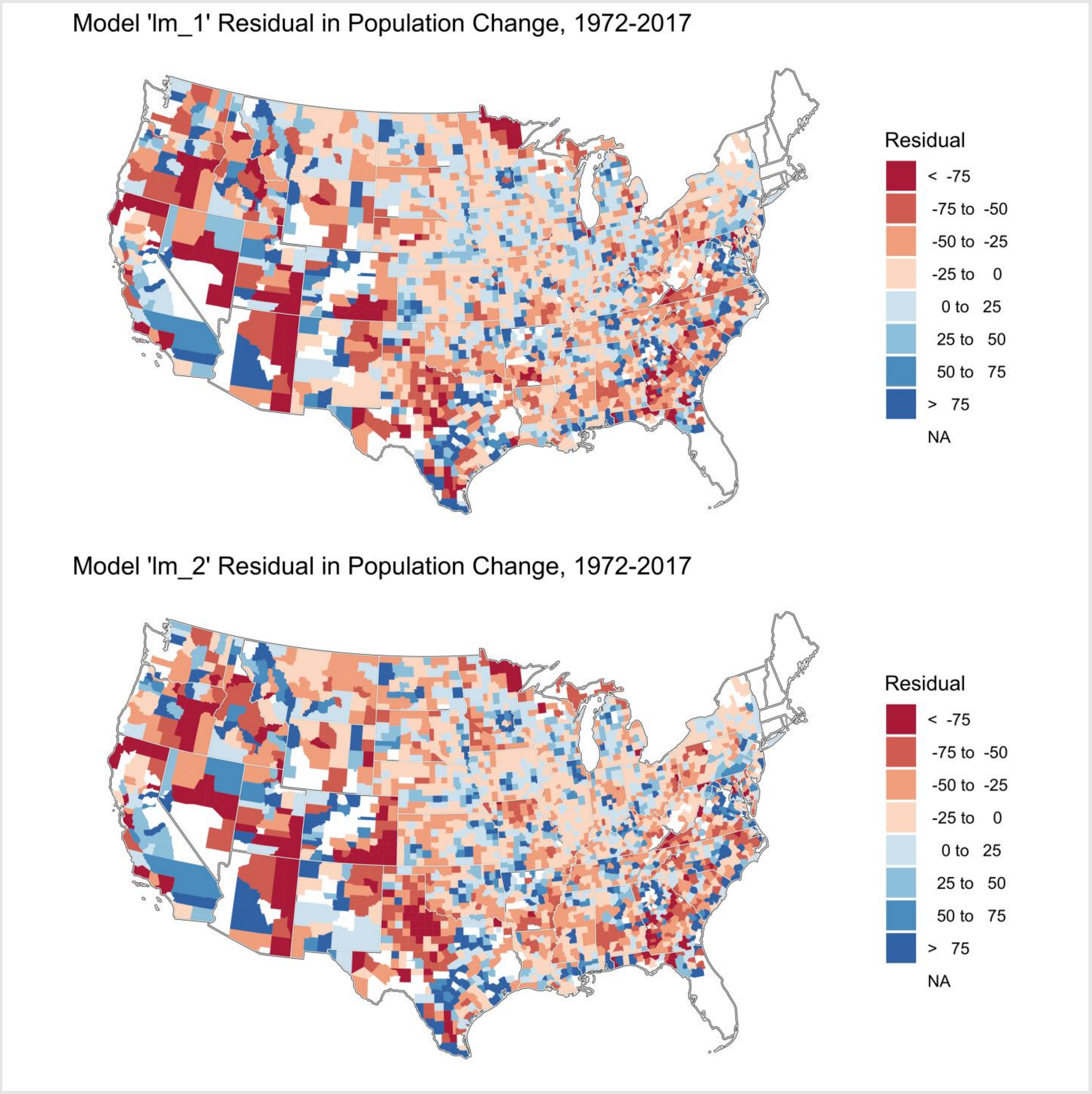


Figure 17. Maps of Model Residuals After Fitting Populating Change with Grain Production Data

These errors on the map (Figure 17) show that the residuals from the two models are qualitatively very similar. Given the fixed effects, the residuals are not concentrated in any particular state. The counties with dark red and dark blue shades are those that experienced particularly large population declines and gains respectively, net the state-level average trends.

In addition to the average effects shown above, we examine how such effects may vary across age groups. To explore this, we first map the population change for two age groups of 15–29 and 60 and older (Figure 18). The first map shows that there are fewer young adults in much of rural America today compared with 1972, particularly in the Great Plains. The second map shows an increase in the elderly population in many parts of the country from 1972 to 2017, except some segments of the Great Plains.

We examine different patterns of associations by applying the previous model to subsets of the data across age groups and time periods. For example, Table 3 presents the results for two age groups (15–29 and 60 and above) and two time periods (1972–1982 and 2002–2017). The variable *ln_grain_prod.lag* is the grain production (in millions of dollars) at the beginning of the time period, and *grain_ch_pct* is the percentage change in grain production during the time period. Two dummy variables are included at the change of -100 percent and 0 percent, for the 2002–2017 data analysis. The results suggest that these effects may be heterogeneous across age groups and time periods.

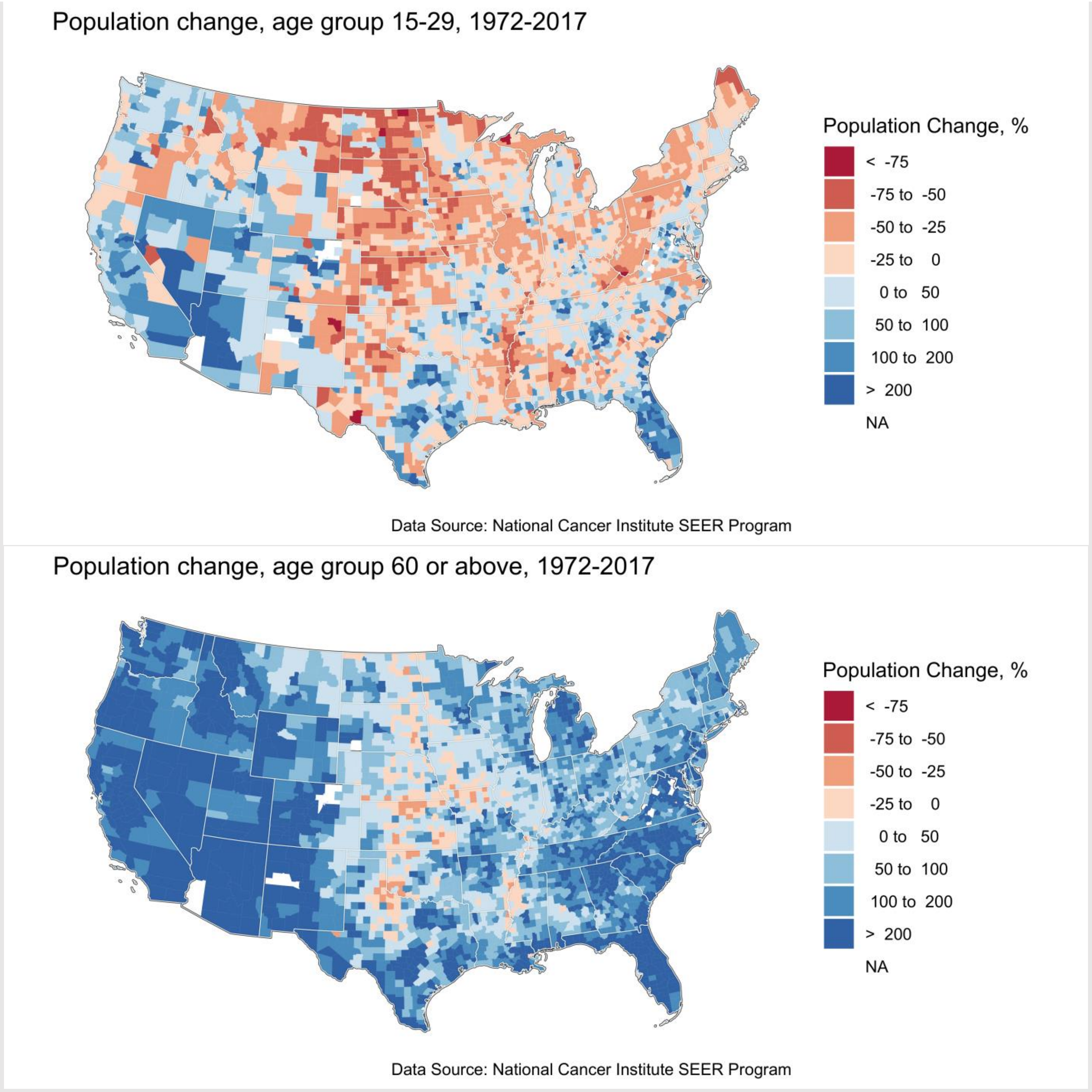

Population change, age group 15-29, 1972-2017
Population Change, %
< -75
-75 to -50
-50 to -25
-25 to 0
0 to 50
50 to 100
100 to 200
> 200
NA
Data Source: National Cancer Institute SEER Program
Population change, age group 60 or above, 1972-2017
Population Change, %
< -75
-75 to -50
-50 to -25
-25 to 0
0 to 50
50 to 100
100 to 200
> 200
NA
Data Source: National Cancer Institute SEER Program

Figure 18. Population Change for Selected Age Group and Time Period

Table 3. Estimate of Grain Production and Population Change for Selected Age Group and Time Period

Variable	Population change, %			
	Models			
	(1)	(2)	(3)	(4)
Log of grain production, 1972	-2.740***	-1.918***	-0.665*	-5.548***
(ln_grain_prod.lag)	(0.273)	(0.251)	(0.342)	(0.593)
Change in grain production	-0.012***	-0.003	-0.011**	-0.016**
(grain_ch_pct)	(0.005)	(0.004)	(0.005)	(0.008)
Indicator for zero grain production			-4.493***	-3.854
(grain_ch_pct == 0)			(1.377)	(2.388)
Indicator for ceased grain production			-1.799	-3.406
(grain_ch_pct == -100)			(1.365)	(2.367)
State fixed effects	Yes	Yes	Yes	Yes
Sample age group	15–29	60 and up	15–29	60 and up
Sample Period	1972–82	1972–82	2002–17	2002–17
Observations	2,719	2,719	2,761	2,761
Adjusted R squared	0.26	0.374	0.111	0.307

Note: Statistical significance $^{*}p < 0.1$; $^{**}p < 0.05$; $^{***}p < 0.01$. Models (1)–(4) are estimated on different subsets of data in terms of age group (15–29 for models (1) and (3): 60 and up for models (2) and (4)) and sample period (1972–1982 for models (1) and (2): 2002–2017) for models (3) and (4).

To analyze such effects systematically, we arrange a grid of subsamples by age group and time period and apply the same estimation model to each subsample. We use five age groups (0–14, 15–29, 30–44, 45–59, 60 and up) and four time periods (1972–1982, 1982–1992, 1992–2002, 2002–2017). In a tibble data frame, which is a special case of the data.frame class, one can split the data by a categorical variable via the function nest() and store such subsets of data in a list-column. We then apply a regression formula to each row of the data-column and store the results in another list-column.

```
# create the age group and time period combination
df_pop_grain <- df_pop_grain %>%
  mutate(age_era = paste0(age_group2, ":", Year,sep = ''))

# create a regression function to be applied to a given data.frame
pop_ch_model <- function(df) {
  lm( pop_ch_pct ~ ln_grain_prod.lag + grain_ch_pct +
         grain_ch_pct_0 + grain_ch_pct_neg100 + St_name, data = df)
}

# function to run a model by group via nest()
run_model_by_group <- function(df, group_var, model_as_function) {
  group_var <- enquo(group_var)
  df2 <- df %>% group_by(!!group_var) %>% nest()
  df2 %>% mutate(
    model = map(data, model_as_function),
    rlt   = map(model, summary) %>% map(coefficients) %>% map(data.frame),
    varname = map(rlt, rownames),
    estimate = map(rlt, ~ .x$Estimate),
    st_error = map(rlt, ~ .x$Std..Error),
    t_stat = map(rlt, ~ .x$t.value)
  )
}

lm_pop_age_era <-
  run_model_by_group(df_pop_grain  %>%
                          filter(!is.na(age_group2), Year >= 1980),
                     group_var = age_era,
                     model_as_function = pop_ch_model)

lm_pop_age_era %>% print(n = 5)
## # A tibble: 20 x 8
## # Groups:   age_era [20]
##   age_era          data model  rlt      varname  estimate st_error t_stat
##   <chr>     <list<df[,43> <list> <list>   <list>   <list>   <list>   <list>
## 1 age_0-14…  [2,719 × 43] <lm>   <df[,4]… <chr [4… <dbl [4… <dbl [4… <dbl …
## 2 age_0-14…  [2,799 × 43] <lm>   <df[,4]… <chr [4… <dbl [4… <dbl [4… <dbl …
## 3 age_0-14…  [2,802 × 43] <lm>   <df[,4]… <chr [4… <dbl [4… <dbl [4… <dbl …
## 4 age_0-14…  [2,761 × 43] <lm>   <df[,4]… <chr [4… <dbl [4… <dbl [4… <dbl …
## 5 age_15-2…  [2,719 × 43] <lm>   <df[,4]… <chr [4… <dbl [4… <dbl [4… <dbl …
## # … with 15 more rows
```

Here, column *data* is a list-column containing different subsets of the data separated by age group-era combination. List-column *model* contains the corresponding regression outputs, which are summarized in another list-column *rlt*, which are further isolated into list-columns of variable names, point estimates, standard errors, and *t* statistics. Each cell in the *estimate* list-column contains a list of coefficient estimates for a given subsample. These coefficient estimates can be extracted by function *unnest()*, which returns a long-format data frame that stacks coefficient estimates for various subsamples according to the age group-era combination.

```
rlt_age_era <- lm_pop_age_era %>%
  select(age_era, varname, estimate, st_error, t_stat) %>%
  unnest(cols = c("varname", "estimate", "st_error", "t_stat"))
rlt_age_era %>% print(n = 5)
## # A tibble: 905 x 5
## # Groups:   age_era [20]
##   age_era       varname          estimate st_error t_stat
##   <chr>         <chr>               <dbl>    <dbl>  <dbl>
## 1 age_0-14:1982 (Intercept)       0.391    2.25     0.174
## 2 age_0-14:1982 ln_grain_prod.lag -2.56     0.282   -9.05
## 3 age_0-14:1982 grain_ch_pct     -0.00711  0.00476 -1.49
## 4 age_0-14:1982 St_nameAR         7.26     3.00     2.42
## 5 age_0-14:1982 St_nameAZ        22.6      5.85     3.86
## # … with 900 more rows
```

For selected coefficients, we summarize the results in Figure 19. The plot on the left shows that people of all ages, the baby boomer generation in particular, moved out of grain-producing rural counties throughout the period spanning 1972–2017. The plot on the right shows that an increase in grain production was associated with a population decline from 1982 to 1992 and post 2002, across age groups.

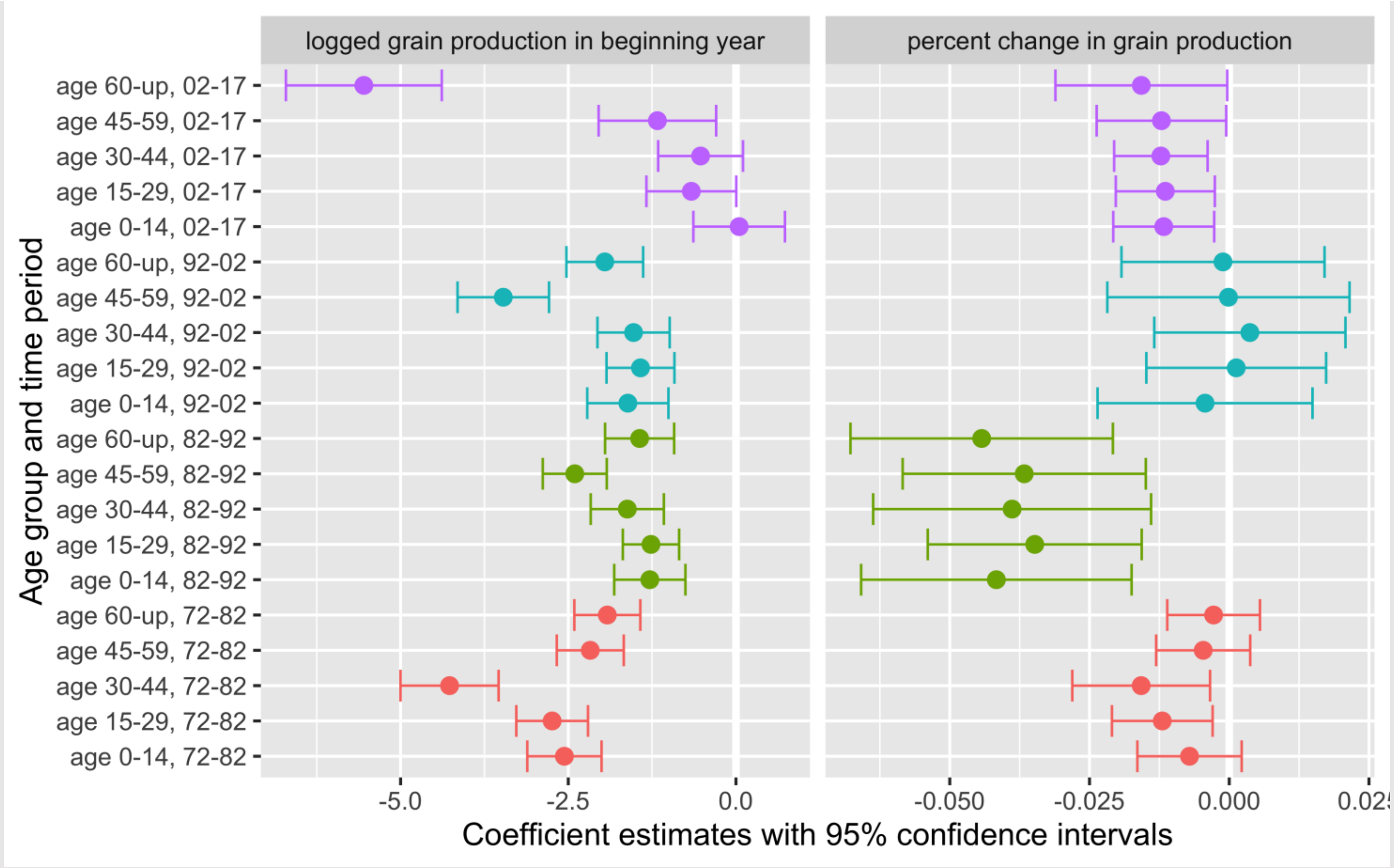


Figure 19. Associations between Grain Production and Population Change by Age Group and Time Period

Note: A OLS regression model is estimated for each subset defined by the combination of age group and time period. The plot shows the coefficient estimates for logged grain production in the beginning of the time period (left) and percentage change in grain production (right).

While the issue of rural depopulation is beyond the scope of our analysis here, it helps to shed light on the associations between grain farming and population change. Many rural communities were initially developed because of the land's potential to produce grain and support the residents. As grain production intensified with time, farms got bigger and fewer, and the communities that relied on grain farming shrunk.

6 Additional Tools

In this section, we briefly describe additional R tools that may be of interest to applied economists.

6.1 *rmarkdown*

The *rmarkdown* package allows for producing documents that combine text, R code, and the output of the code all in one place. It also accommodates LaTex math symbols and equations. Its output can be produced in several file types such as HTML, PDF, and Microsoft Word. *rmarkdown* can be useful for taking notes during data analyses, preparing lab reports, or drafting technical manuscripts. A template is available in RStudio Integrated Development Environment (IDE).

6.2 *flexdashboard*

As a special case of *rmarkdown* document, the *flexdashboard* output class allows one to easily assemble a dashboard-style layout consisting of separate segments of output panes. For example, multiple plots and tables can be arranged in columns and rows all in one screen. A *flexdashboard* template is available in RStudio IDE.

6.3 *shiny*

With *shiny* package, one can develop interactive applications that can run on local computers or be deployed online. A template is available in RStudio IDE. To learn more, a good place to start is a tutorial by RStudio.[12]

6.4 *dygraphs*

With the *dygraphs* package, one can create interactive time-series plots on which the user can see values associated with selected data points with mouse-over actions and select a time pan of the plot to zoom in and out. Here is a simple example that is plotted in Figure 20:

```r
library(dygraphs)
load(file="ts_milk_price.RData")

PA <- ts_milk_price %>% filter(state_alpha == "PA") %>%
  select(Value) %>%
  ts(start = c(1990, 1), end = c(2019, 08), frequency = 12)

CA <- ts_milk_price %>% filter(state_alpha == "CA") %>%
  select(Value) %>%
  ts(start = c(1990, 1), end = c(2019, 08), frequency = 12)

cbind(PA, CA) %>%
  dygraph(main = "Monthly Milk Price, $/cwt") %>%
  dyRangeSelector()
```

[12] https://shiny.rstudio.com/

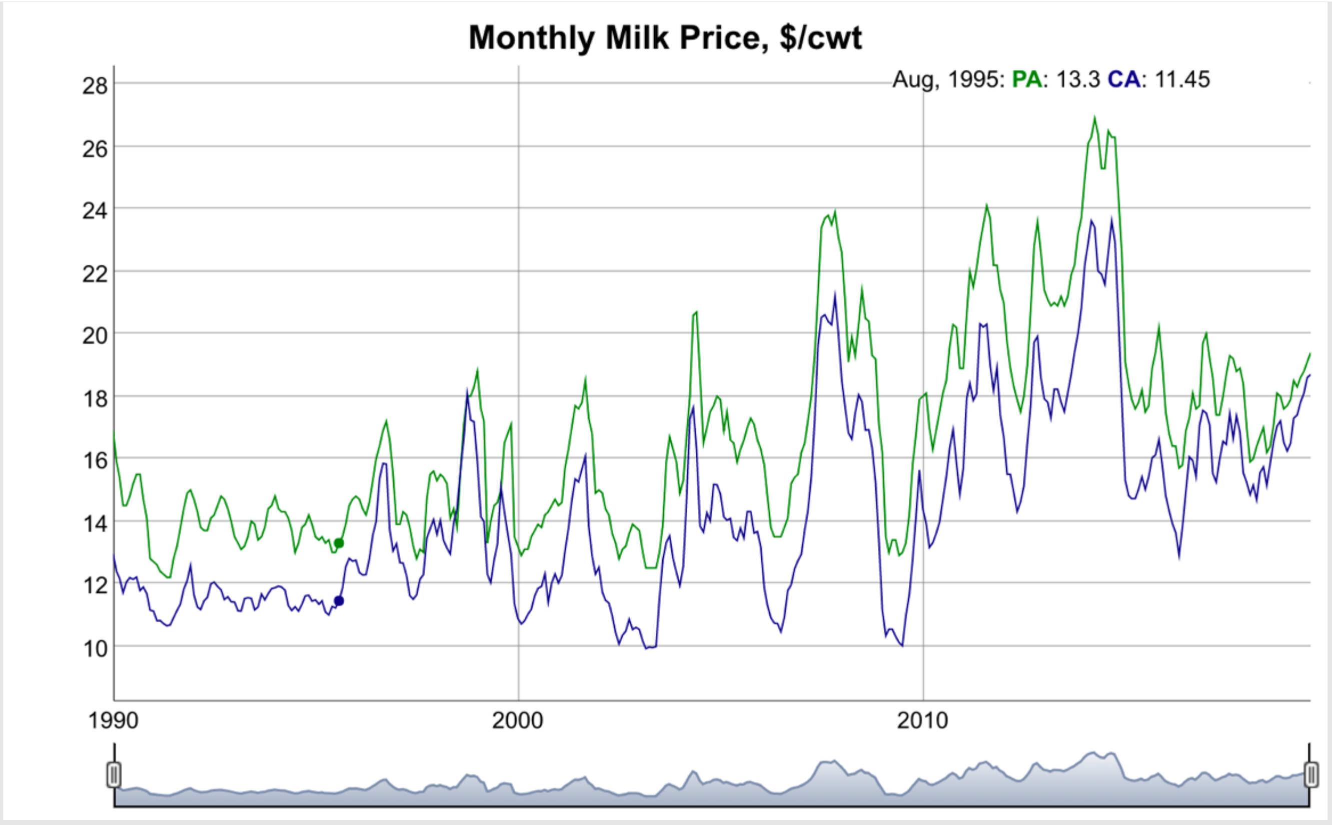


Figure 20. Example of an Interactive Dygraphs Plot for Pennsylvania and California Monthly Milk Prices

6.5 *leaflet*

The leaflet package lets one create interactive maps that can be hosted online with base maps provided by OpenStreetMap and CartoDB. Figure 21 was developed with data from the U.S. Agricultural Census to show the distribution of farms across the conterminous United States that reported using value-added marketing methods in 2017.

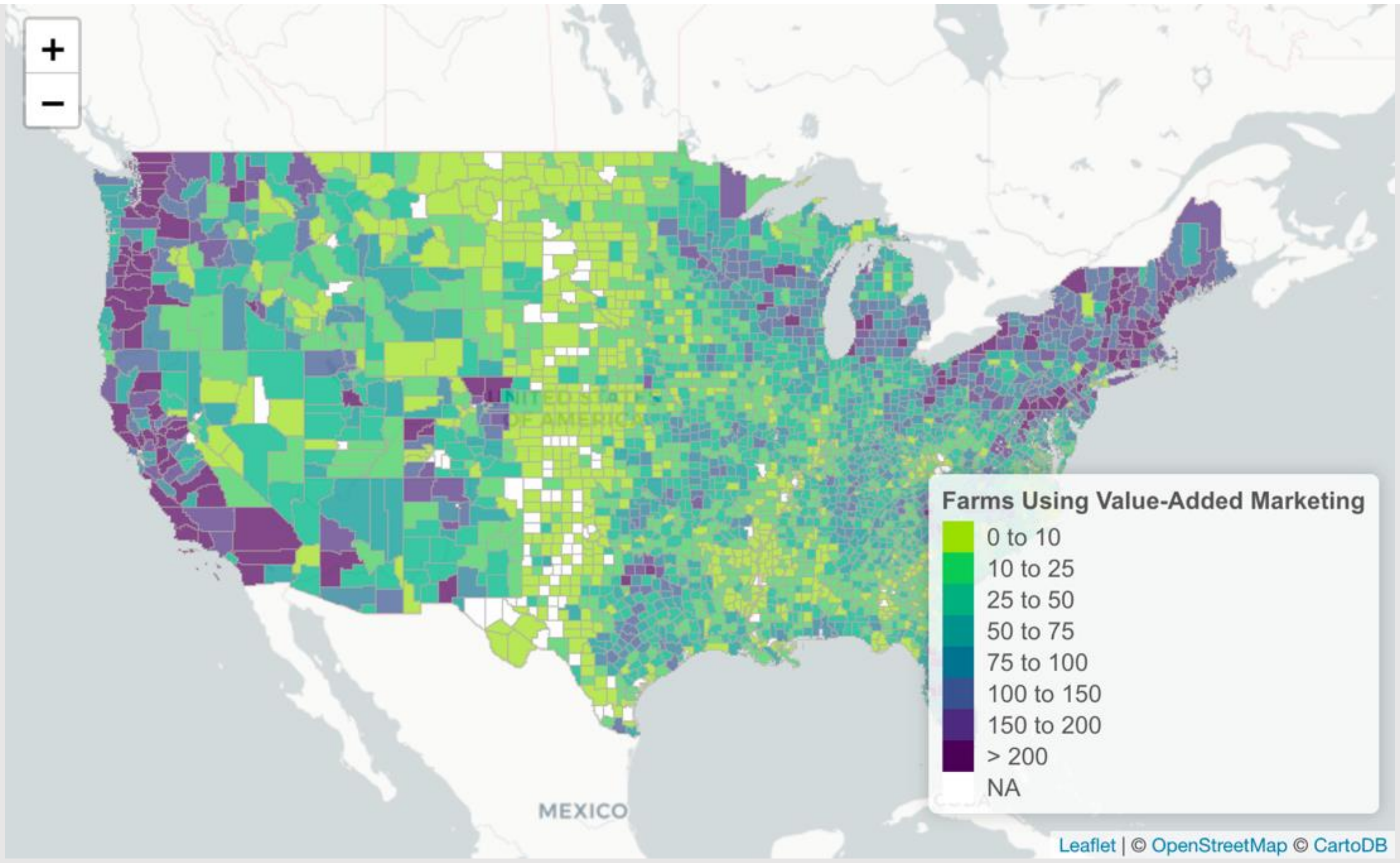


Figure 21. Example of an Interactive Leaflet Map That Allows for Zooming In or Out and Selecting the Area in View

6.6 Cheatsheets

We recommend all readers to explore a collection of cheatsheets hosted by RStudio.[13] The cheatsheets provide great summaries of popular R packages and their examples. R beginners would find the cheatsheets about R-programming basics and RStudio IDE useful. Experienced R users may encounter recently uploaded and noteworthy packages for popular topics such as big data management, machine-learning, and integration with other programming environments.

6.7 Online Searches

R users quickly learn that the best way to find programming information or to get help is through online searches. A keyword search usually turns up relevant online Q&A discussions, which work remarkably well for troubleshooting (e.g., with fine-tuning data plots).

6.8 *data.table*

Although this article focuses on the *dplyr* package for data transformation, a popular alternative is the *data.table* package. For example, the following code performs the parallel tasks with some of the *dplyr* code we presented above (i.e., selecting the census table that contains the number of farms by farm sales class and also aggregating them into a binary farm sales class). Note the differences in the syntax of the two packages.

The reader may find that the syntax of *data.table* is not as readable as that of *dplyr*. Indeed, the developer of *dplyr* intentionally designed its syntax to be easy to read. Interested readers may be referred

[13] https://rstudio.com/resources/cheatsheets/

to [online discussions](https://stackoverflow.com/questions/21435)[14] or [side-by-side comparisons.](https://atrebas.github.io/post/2019-03-03-datatable-dplyr/)[15] Also, notice the use of the same piping operator *%>%*, which in fact belongs to the *magrittr* package (from which *dplyr* imports it). An advantage

```r
library(data.table)
library(magrittr)

us17_dt <- data.table(us17)
us17_dt[census_table == 2 &
      grepl("COMMODITY TOTALS - OPERATIONS WITH SALES", Item) &
      !is.na(Class),
     c("Class", "Value")]

county17_dt <- data.table(county17)
county17_dt[
    census_table == 2 &
    grepl("COMMODITY TOTALS - OPERATIONS WITH SALES", Item) &
    !is.na(Class) & Co_name! ="NULL",
    class_S_NS : = ifelse(Class %in% class_S, "S", "NS")] %>%
  .[, .(Value = sum(Value, na.rm = T)),
    by = c("St_code", "St_name", "Co_code", "Co_name", "class_S_NS")] %>%
  .[class_S_NS =="S"] %>%
  .[order(-Value)] %>% head(n = 10)
```

of *data.table* over *dplyr* is its computational speed, which can become important for large data sets (say, greater than 1 GB). For those who prefer the *dplyr* syntax but want the speed of *data.table*, try a package called *dtplyr*, which is currently being developed by the developer of *dplyr* package as a *data.table* backend for *dplyr.*[16]

6.9 *sparklyr*

Recent progress in the R and Spark integration now enables one to use R for processing so-called big data (e.g., in a distributed data file system like Apache Hadoop or in a streaming data platform like Apache Kafka). With the *sparklyr* package,[17] one can combine the core EDA techniques through the *dplyr* and *ggplot2* packages with large-scale data processing in Apache Spark, without holding the data in the local machine's memory. Put simply, *sparklyr* connects an R session with Spark, translates *dplyr* functions into Hive SQL code, and submits the code to the Spark connection. One can read a subset of data or data summary, generated by such *dplyr* data transformations, into the local machine's memory by the *collect()* function for data visualization by *ggplot2* . Moreover, the *sparklyr* package provides additional functions to utilize Spark's machine-learning library APIs, integrate a shiny application with big data, and build a data pipeline (e.g., a sequence of data cleaning, transformation, modeling, and prediction), which can be further exported as an API using the *mleap* package.

[14] https://stackoverflow.com/questions/21435.
[15] https://atrebas.github.io/post/2019-03-03-datatable-dplyr/.
[16] available from https://github.com/tidyverse/dtplyr.
[17] One can practice many functionalities of the `sparklyr` package with a simple local installation of Spark, without any access to an actual big data connection. For more information, see https://spark.rstudio.com/ and https://therinspark.com/.

7 Concluding Remarks

We have reviewed the core tools of data visualization and exploration from the recent developments in R freeware. We believe this new generation of tools would be a great asset for economists and students in applied economics. Hands-on learning with such tools can be highly complementary to many of economics courses, and given today's high demand for data scientists, it is valuable for students to acquire practical skills for EDA. In addition to their knowledge of statistics and econometrics, many students would be empowered to learn how to explore real-world data and become capable of generating effective data narratives and new hypotheses.

To advance students' skills in data analyses and cultivate their interests in economic issues, we suggest three directions of future efforts. First, teaching examples and case studies on EDA education may be shared through teaching journals, like this publication. Second, to aid instructors who undertake such teaching, applied economics departments may dedicate some tutorial hours for EDA and hire experienced students as peer tutors. Third, applied economics conferences may host undergraduate competitions for data visualization projects, which focus on public education and outreach rather than research outputs. On the last point, the hurdle for creating data visualization materials or data narratives is much lower, compared to producing new research findings, and therefore such projects will be able to engage a larger body of students. While it may not be called research in itself, the creation of insightful data plots can contribute to public knowledge, and hence it would merit recognition in applied economics communities. Through the combination of hands-on-learning, technical support, and academic recognition, EDA education can be made an integral part of an applied economics curriculum.

About the Author: Kota Minegishi is an Assistant Professor at the University of Minnesota, Twin Cities (corresponding author: kota@umn.edu). Taro Mieno is an Assistant Professor at the University of Nebraska-Lincoln.

References

Alonzo, A. 2016. "Top 5 Broiler Producers Dominate US Production."Retrieved from https://www.wattagnet.com/articles/26925-top-5-broiler-producers-dominate-us-production

Athey, S., J. Tibshirani, andS. Wager. 2019. "Generalized Random Forests."The Annals of Statistics47(2):1148–1178.

Coble, K.H., A.K. Mishra, S.Ferrell, andT. Griffin. 2018. "Big Data in Agriculture: A Challenge for the Future."Applied Economic Perspectives and Policy40(1):79–96.

Healy, K. 2018. Data Visualization: A Practical Introduction, 1sted. Princeton NJ: Princeton University Press.

Ismay, C., and A.Y. Kim. 2019. Statistical Inference via Data Science: A ModernDive into R and the Tidyverse,1sted. Boca Raton: Chapman and Hall/CRC.

Johnson, K.M., andG.V. Fuguitte. 2000. "Continuity and Change in Rural Migration Patterns, 1950–1995."Rural Sociology65(1):27–49.

Kabacoff, R. 2018. Data Visualization with R.Online open-source book accessed athttps://rkabacoff.github.io/datavis/

Longworth, R.C. 2009. Caught in the Middle: America's Heartland in the Age of Globalism.New York: Bloomsbury USA.

Lovelace, R., J. Nowosad, and J. Muenchow.2019. Geocomputation with R, 1sted. Boca Raton: ChapmanandHall/CRC.

O'Donoghue, E., R. Hoppe,D.Banker, and P. Korb. 2009. Exploring Alternative Farm Definitions: Implications for Agricultural Statistics and Program Eligibility. Economic Information Bulletin No. 49. Washington DC: U.S. Department of Agriculture.

Storm, H., K. Baylis, and H. Heckelei. 2019. "Machine Learning in Agricultural and Applied Economics." European Review of Agricultural Economics. https://doi.org/10.1093/erae/jbz033

Twain, M. 1892. The American Claimant. New York: Charles L. Webster.

Walzer, N. 2003. The American Midwest: Managing Change in Rural Transition,1sted. Armonk NY: Routledge.

White, K.J.C. 2008. "Population Change and Farm Dependence: Temporal and Spatial Variation in the U.S. Great Plains, 1900–2000."Demography45(2):363–386.

Wickham, H., and G. Grolemund.2017. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data, 1sted. Sebastopol CA: O'Reilly Media.

Wickham, H., M. Averick, J.Bryan, W.Chang, L.McGowan, R. François, G.Grolemund, . . .H. Yutani. 2019. "Welcome to the Tidyverse." Journal of Open Source Software4(43):1686.

Wilkinson, L. 2005. The Grammar of Graphics.Springer.

Wood, D. 2018. "Costco Poultry Processing Plant to Boost Nebraska Economy."Retrieved from https://www.acppubs.com/articles/7398-costco-poultry-processing-plant-to-boost-nebraska-economy

2(3) doi: 10.22004/ag.econ.303913