Session - 6

Merging with more than one variable

- Most of the merging usage is with two variables: date and company name.

Merging with more than one variable

- Most of the merging usage is with two variables: date and company name. Generate data using below:

```
date = seq.Date(as.Date("2018-01-01"), by = 1, length.out = 5);

comp = c("A", "B", "C", "D", "E");

all_pairs = merge(comp, date, by = NULL);

# sales data - 15 points
set.seed(1);

idx = sample(1:nrow(all_pairs), 15, replace = F);

df1 = data.frame(comp = all_pairs$x[idx], date = all_pairs$y[idx],
                 sales = round(runif(15, min = 1e3, max = 1e5)));

# advertising data - 12 points

idx = sample(1:nrow(all_pairs), 12, replace = F);

df2 = data.frame(comp = all_pairs$x[idx], date = all_pairs$y[idx],
                 adv = round(runif(12, min = 1e2, max = 1e4)));
```

df1

comp	date	sales
B	05-04-2018	53429
A	03-04-2018	70750
B	03-04-2018	5426
B	06-04-2018	88504
E	06-04-2018	78449
A	04-04-2018	98954
D	04-04-2018	57618
E	04-04-2018	22011
C	02-04-2018	85976
D	02-04-2018	42842
E	02-04-2018	94057
A	05-04-2018	25063
E	05-04-2018	51320
D	06-04-2018	99720
E	03-04-2018	16845

df2

comp	date	adv
C	03-04-2018	980
C	06-04-2018	6183
B	06-04-2018	8077
B	02-04-2018	241
D	02-04-2018	3642
B	04-04-2018	3878
E	02-04-2018	1501
B	03-04-2018	6727
D	06-04-2018	4259
D	03-04-2018	4434
A	06-04-2018	6745
E	03-04-2018	2208

Inner Join

- merge(df1, df2, by = c(" comp", "date"))

comp	date	sales	adv
B	03-04-2018	5426	6727
B	06-04-2018	88504	8077
D	02-04-2018	42842	3642
E	02-04-2018	94057	1501
D	06-04-2018	99720	4259
E	03-04-2018	16845	2208

Full Join

- merge(df1, df2, by = c(" comp", "date"), all = T)

comp	date	sales	adv
B	05-04-2018	53429	NA
A	03-04-2018	70750	NA
B	03-04-2018	5426	6727
B	06-04-2018	88504	8077
E	06-04-2018	78449	NA
A	04-04-2018	98954	NA
D	04-04-2018	57618	NA
E	04-04-2018	22011	NA
C	02-04-2018	85976	NA
D	02-04-2018	42842	3642
E	02-04-2018	94057	1501
A	05-04-2018	25063	NA
E	05-04-2018	51320	NA
D	06-04-2018	99720	4259
E	03-04-2018	16845	2208
C	03-04-2018	NA	980
C	06-04-2018	NA	6183
B	02-04-2018	NA	241
B	04-04-2018	NA	3878
D	03-04-2018	NA	4434
A	06-04-2018	NA	6745

Left Join

- `merge(df1, df2, by = c(" comp", "date"), all.x = T)`

comp	date	sales	adv
B	05-04-2018	53429	NA
A	03-04-2018	70750	NA
B	03-04-2018	5426	6727
B	06-04-2018	88504	8077
E	06-04-2018	78449	NA
A	04-04-2018	98954	NA
D	04-04-2018	57618	NA
E	04-04-2018	22011	NA
C	02-04-2018	85976	NA
D	02-04-2018	42842	3642
E	02-04-2018	94057	1501
A	05-04-2018	25063	NA
E	05-04-2018	51320	NA
D	06-04-2018	99720	4259
E	03-04-2018	16845	2208

Right Join

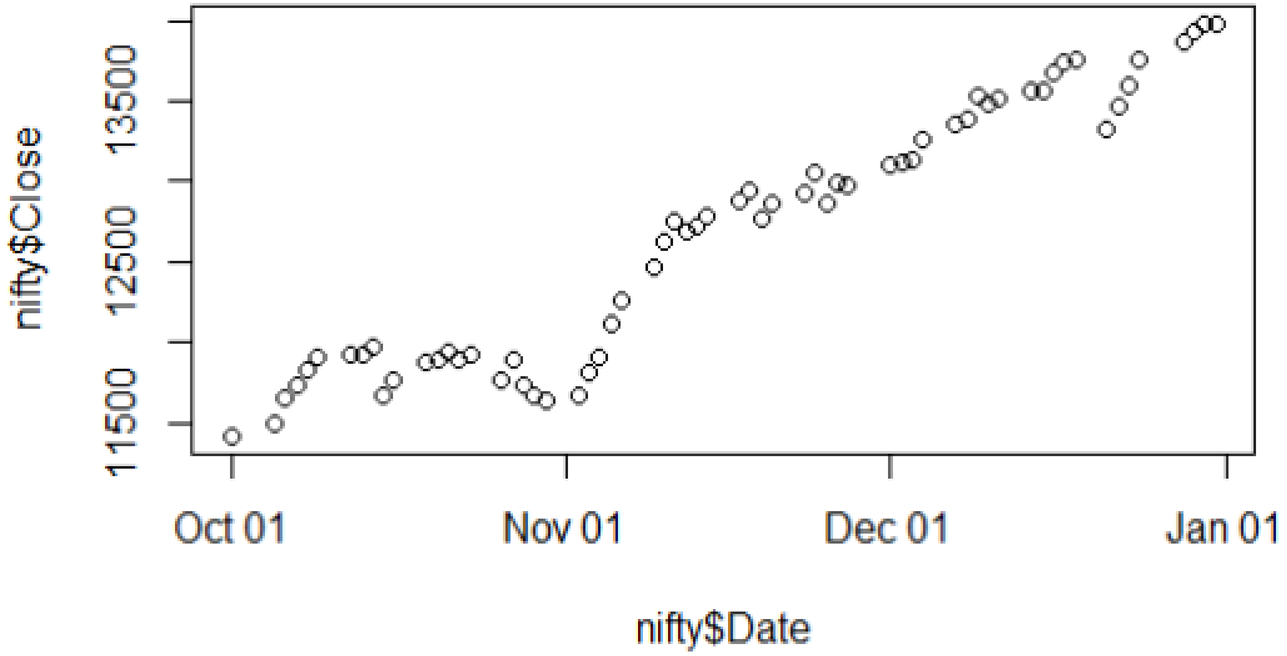
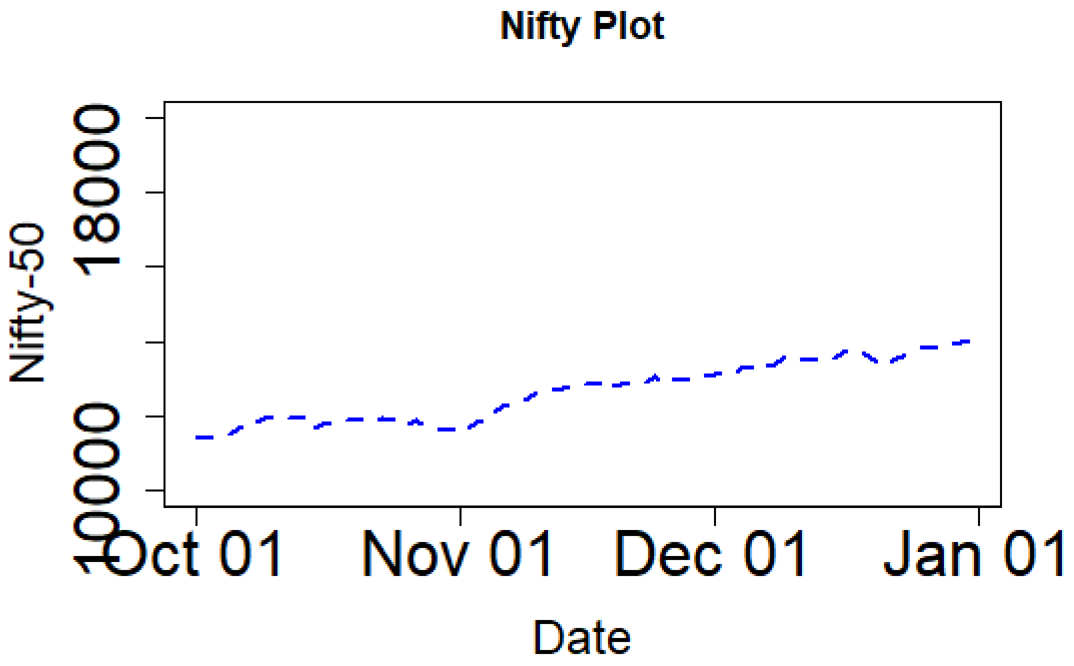
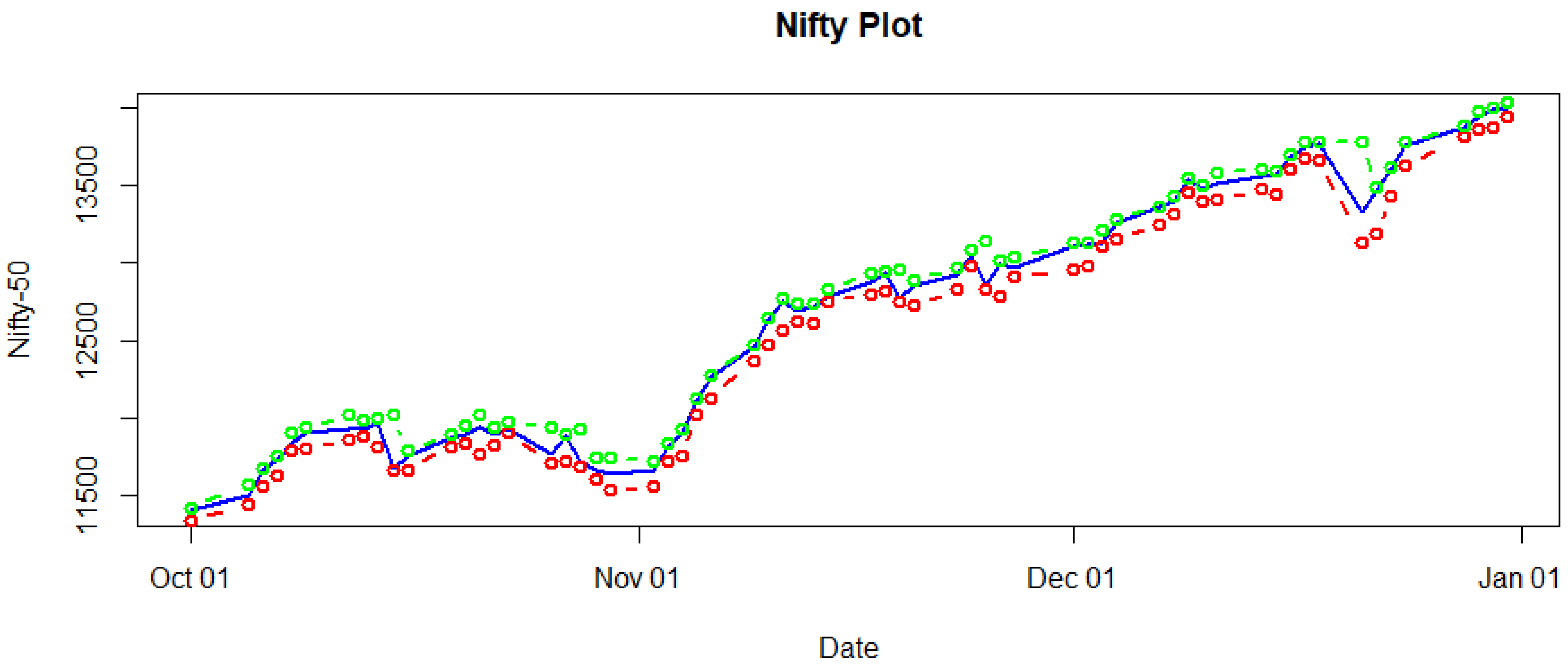
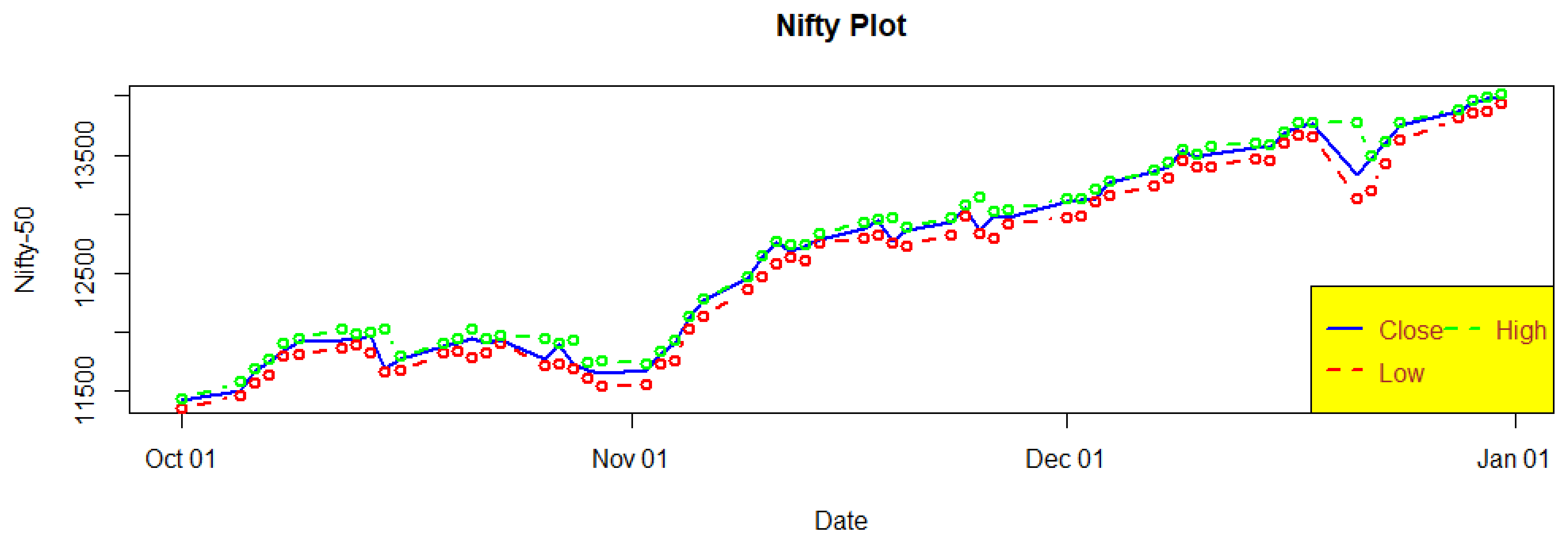
- `merge(df1, df2, by = c(" comp", "date"), all.y = T)`

comp	date	sales	adv
C	03-04-2018	NA	980
C	06-04-2018	NA	6183
B	06-04-2018	88504	8077
B	02-04-2018	NA	241
D	02-04-2018	42842	3642
B	04-04-2018	NA	3878
E	02-04-2018	94057	1501
B	03-04-2018	5426	6727
D	06-04-2018	99720	4259
D	03-04-2018	NA	4434
A	06-04-2018	NA	6745
E	03-04-2018	16845	2208

Anti Join

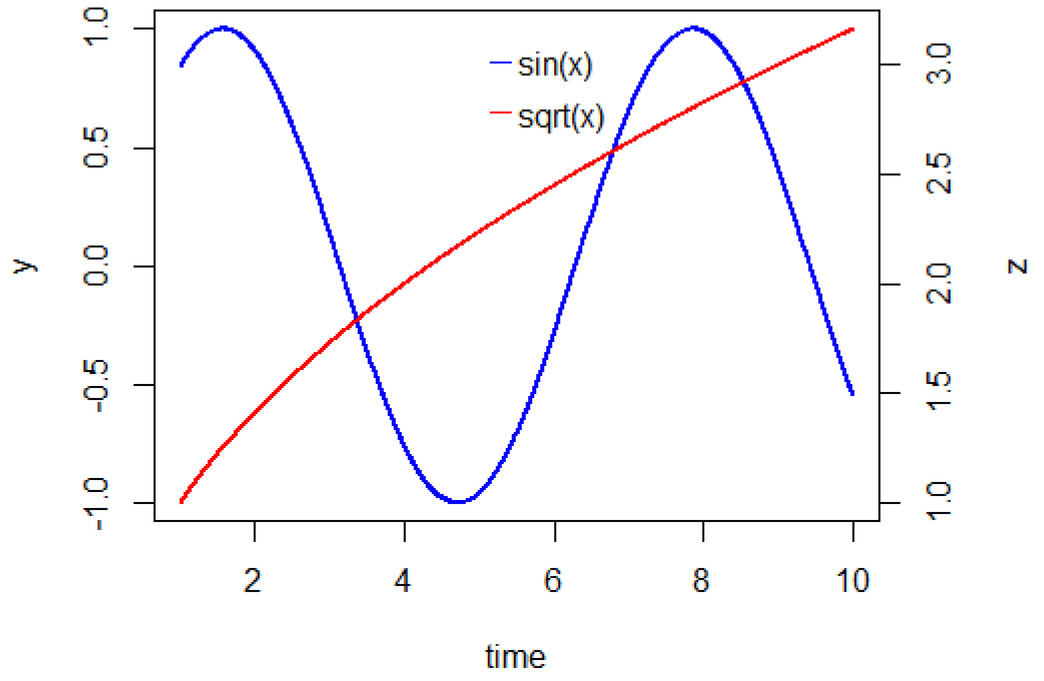
- Sometimes, we wish to find observations in `df1` that are not in `df2`
- `setkey(df1, comp, date); setkey(df2, comp, date);`
- `df1[!df2]` is anti-join.
  - `df2[!df1]` is also anti-join (but the other way round)
- `df1[df2]` is the observations in `df1` that are also in `df2`. This is same as `merge(df1, df2, all.y = T); # right-join`
- `df2[df1]` is left-join. `df1[df2, nomatch = 0]` is inner-join.
- There is no short-hand for full outer join and Cartesian product.
- We can't do anti-join using `merge()` commands.

Reshaping (long to wide and vice-versa)

- Example data (long form):
  - `dt = fread("long_form_returns.csv", na.strings = "");`
  - Four columns: `cusip, Date, retadj, industry`
- Suppose, we want each firm (`cusip`) in a separate column
  - `wide.ret = dcast(dt, Date + industry ~ cusip, value.var = "retadj");`
    - The wide-form variable comes after ~
    - Long-form variables comes before ~
    - Finally, we want “returns” as the value of wide-from format
- How do we get the long form back from `wide.ret`?
  - ```
    dt.org = melt(wide.ret,
                  id.vars = c("Date", "industry"),
                  measure.vars = 3:ncol(wide.ret),
                  variable.name = "cusip", value.name = "retadj",
                  variable.factor = F, na.rm = T);
    ```
    ```

- Wide-form data has only one “value” variable:
  - Rows are dates and columns are company names and the matrix inside is returns. It’s very difficult to get more than one variable.
    - Try: `dt[, ret2 := retadj];`
      `dcast(dt, Date + industry ~ cusip, value.var = c("retadj", “ret2"));`
    - The column names take the form: `retadj_<cusip>` and `ret2_<cusip>`
    - N cusips, D dates and M other variables (like `retadj`, `ret2` etc)
      - long-form: N*D rows of M variables
      - wide-form: D rows of N*M variables

- If there are multiple matches in converting to wide-form
  - We can aggregate data using any function like `sum`, `mean`, …
  - `wide.ret = dcast(dt, Date ~ industry, value.var = "retadj", fun.aggregate = mean, na.rm = T);`
  - This step is irreversible in the sense that we can’t get back original data

- Most of the data will be in long-form but you should know how to quickly convert wide-form to long-form.
  - World bank provides data in wide-form

Plotting in R

Plotting in R

- `plot(x, y, <OPTIONS>);`
- `plot(nifty$Date, nifty$Close);`

Plotting in R

- `plot(x, y, <OPTIONS>);`
- `plot(nifty$Date, nifty$Close);`
- The below are equivalent to the plot command above:
  - `plot(Date, Close, data = nifty);`
  - `nifty[, plot(Date, Close)]; # only if nifty is a DT`

Plotting in R

- `plot(x, y, <OPTIONS>);`
- `plot(nifty$Date, nifty$Close);`
- The below are equivalent to the plot command above:
  - `plot(Date, Close, data = nifty);`
  - `nifty[, plot(Date, Close)]; # only if nifty is a DT`



Plotting Options

```
plot(nifty$Date, nifty$Close,
     type = 'l',
     col = "blue",
     lty = 2,
     lwd = 2,
     xlab = "Date",
     ylab = "Nifty-50",
     main = "Nifty Plot",
     cex.axis = 2,
     cex.lab = 1.5,
     ylim = c(1e4, 2e4));
```



Multiple Lines

- `plot(nifty$Date, nifty$Close, type = 'l', col = "blue", lty = 1, lwd = 2, xlab = "Date", ylab = "Nifty-50", main = "Nifty Plot");`
- `lines(nifty$Date, nifty$Low, type = 'b', col = "red", lty = 2, lwd = 2);`
- `lines(nifty$Date, nifty$High, type = 'b', col = "green", lty = 2, lwd = 2);`



Legend

- ```
  legend("bottomright", legend = c("Close", "Low", "High"),
         col = c("blue", "red", "green"),
         lty = c(1,2,2), lwd = 2,
         bg = "yellow", text.col = "brown", ncol = 2)
  ```


  ```

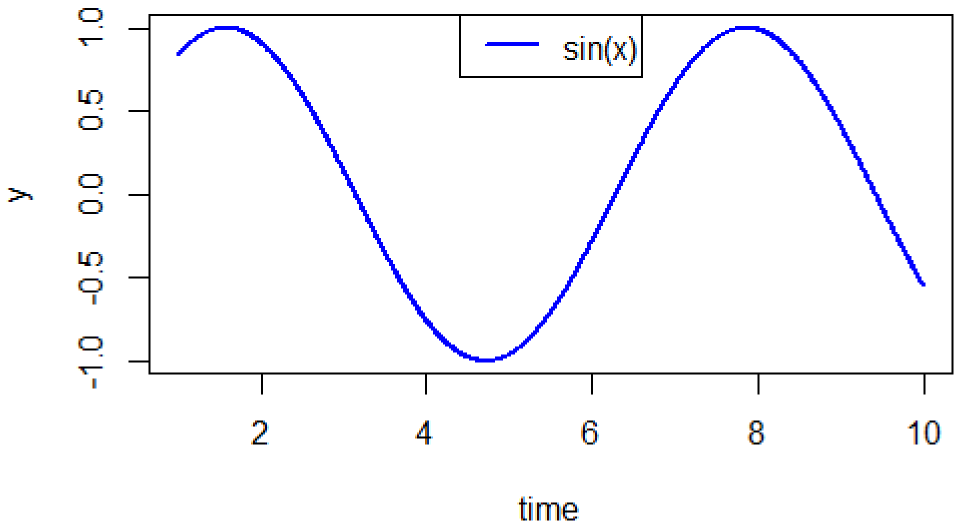
Multiple Axes

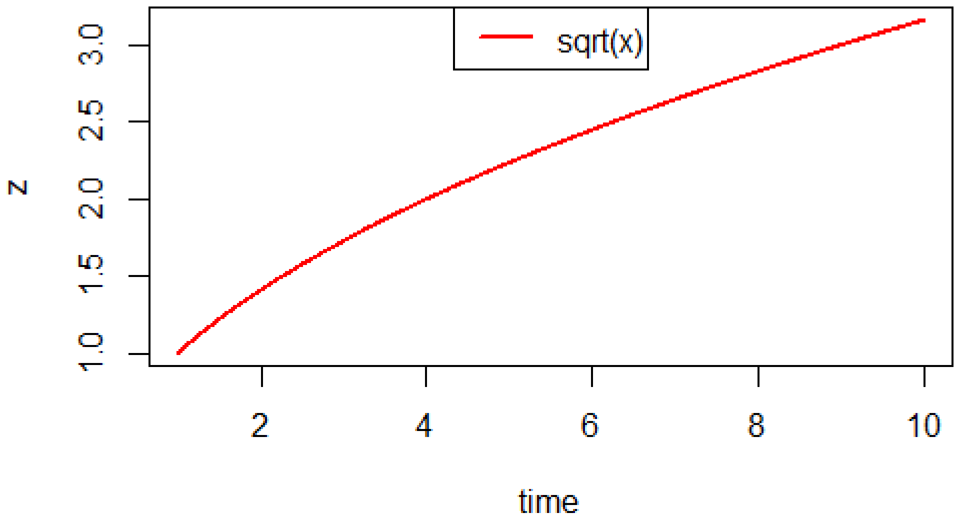
```
x = seq(1, 10, length.out = 1e3);
y = sin(x);
z = sqrt(x);

org_mar = par("mar");
par(mar = c(5,5,2,5)) # for extra margin on right y-axis

plot(x, y, type = "l", col = "blue", lwd = 2, xlab = "time", ylab = "y");
par(new = TRUE);

plot(x, z, type = "l", col = "red", lwd = 2, xlab = NA, ylab = NA, axes = F);

axis(4); # makes axis on RHS
mtext(side = 4, line = 3, "z"); # RHS text

legend("top", legend = c("sin(x)", "sqrt(x)"), lty = 1, col = c("blue", "red"), lwd = 2, bg = NULL);

par(mar = org_mar);
```



Multiple Plots

```
x = seq(1, 10, length.out = 1e3);
y = sin(x);
z = sqrt(x);

org_mfrow = par("mfrow");
par(mfrow = c(2,1));

plot(x, y, type = "l", col = "blue", lwd = 2, xlab = "time",
ylab = "y");
legend("top", legend = c("sin(x)"), lty = 1, col = c("blue"),
lwd = 2);

plot(x, z, type = "l", col = "red", lwd = 2, xlab = "time",
ylab = "z");
legend("top", legend = c("sqrt(x)"), lty = 1, col = c("red"),
lwd = 2);

par(mfrow = org_mfrow);
```





Regression Basics

Regression Basics

- Let, $Y = \beta_0 + \beta_1 \cdot X + u$ be the true model. By regressing Y on X, we hope to recover an unbiased estimate of β_1 and see how much of the variation in Y is explained by variation in X unrelated to variation in u.

Regression Basics

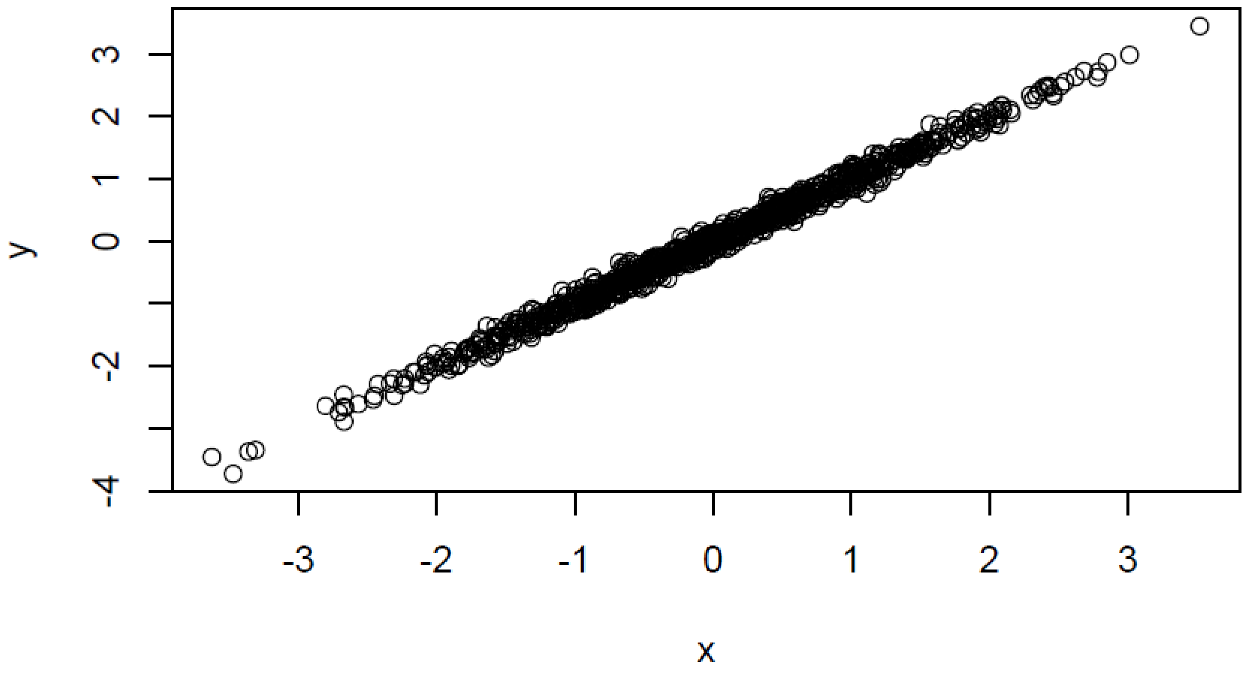
- Let, $Y = \beta_0 + \beta_1 \cdot X + u$ be the true model. By regressing Y on X, we hope to recover an unbiased estimate of β_1 and see how much of the variation in Y is explained by variation in X unrelated to variation in u.

```
n = 1000;
x = rnorm(n, 0, 1);
u = rnorm(n, 0, 0.1);
y = u + x;
plot(x,y);
```

Regression Basics

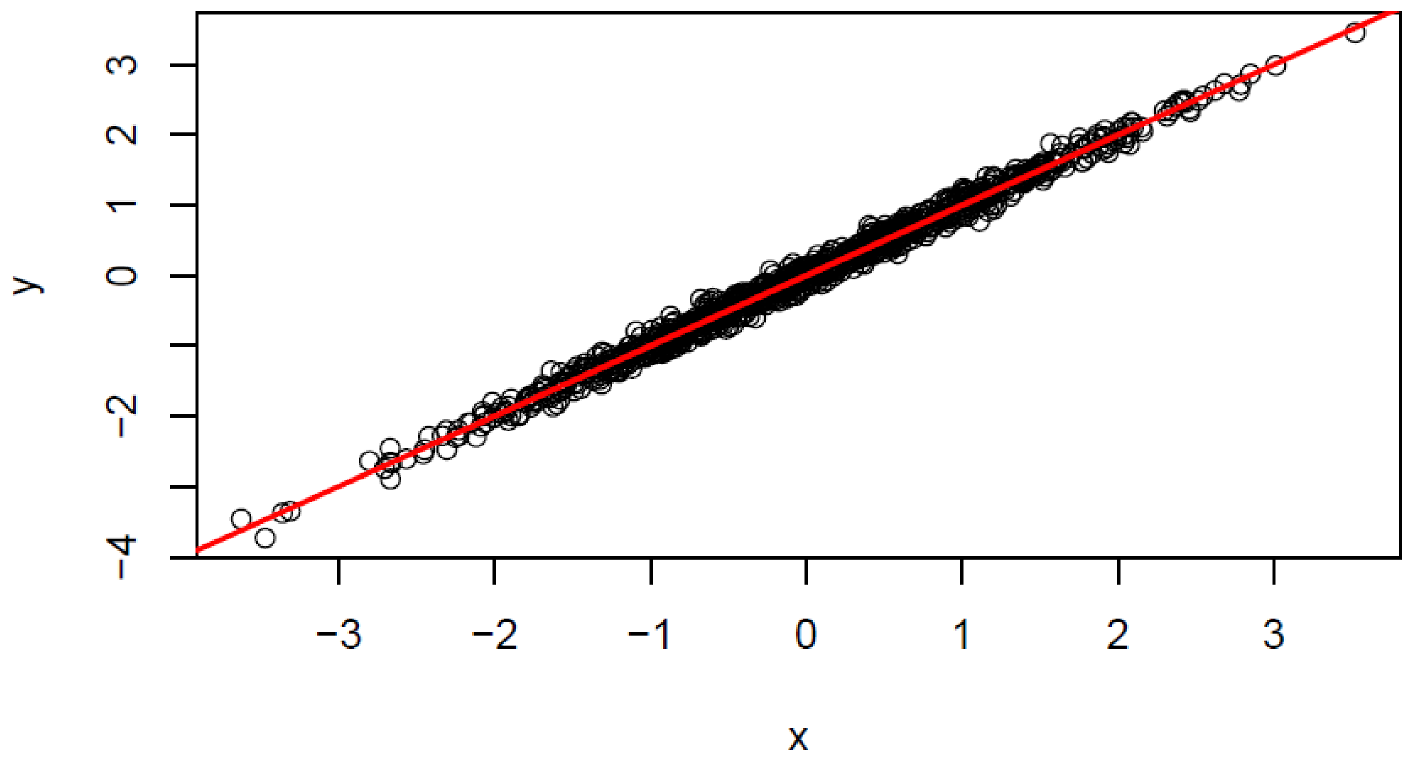
- Let, $Y = \beta_0 + \beta_1 \cdot X + u$ be the true model. By regressing Y on X, we hope to recover an unbiased estimate of β_1 and see how much of the variation in Y is explained by variation in X unrelated to variation in u.

```
n = 1000;
x = rnorm(n, 0, 1);
u = rnorm(n, 0, 0.1);
y = u + x;
plot(x,y);
```

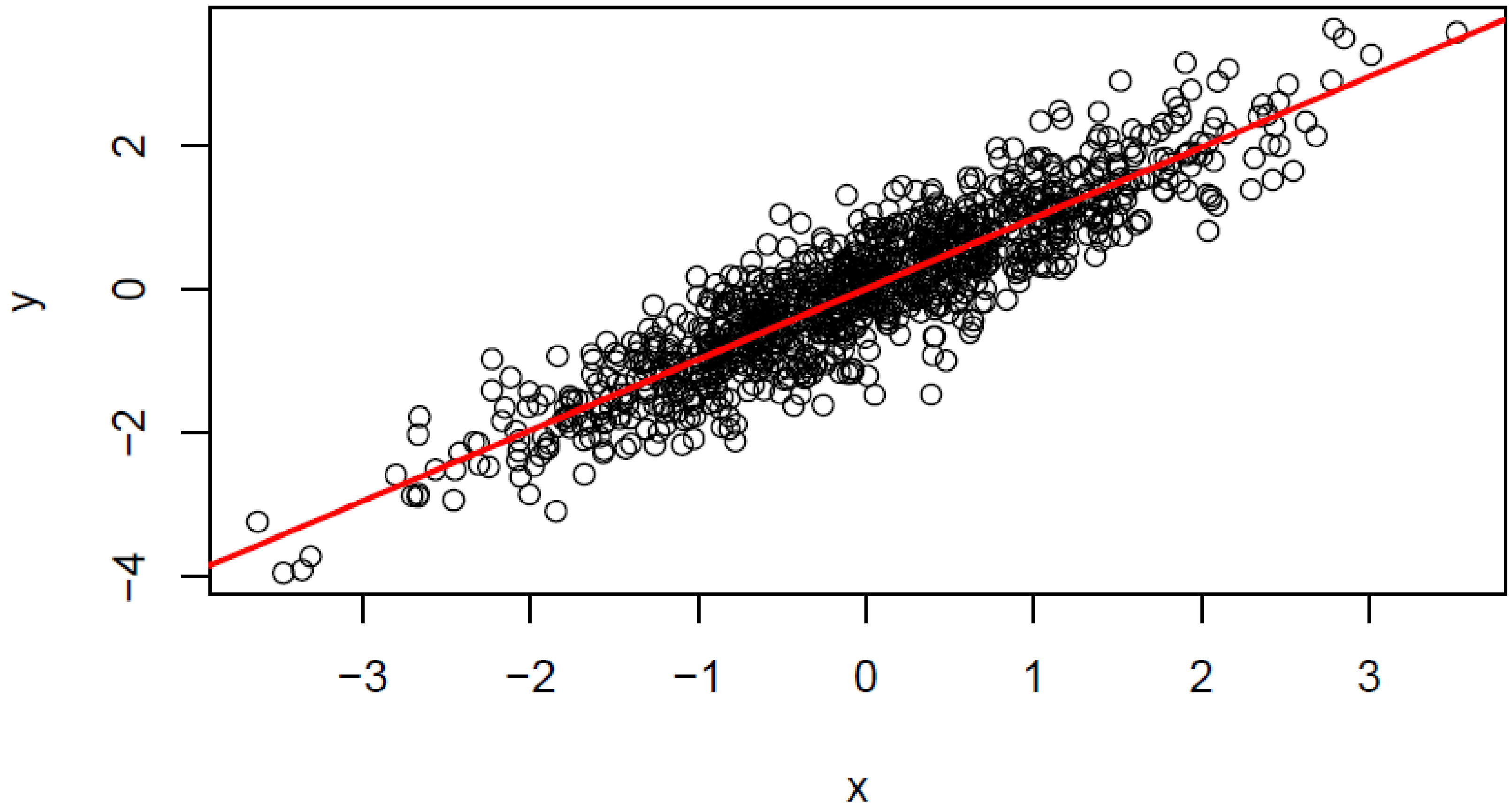


```
fit = lm(y ~ x);
summary(fit);
stargazer(fit, type = "html", out = "fit.html");
# Regression Line
abline(fit$coefficients, col = "red", lwd = 2);
```

```r
fit = lm(y ~ x);
summary(fit);
stargazer(fit, type = "html", out = "fit.html");
# Regression Line
abline(fit$coefficients, col = "red", lwd = 2);
```

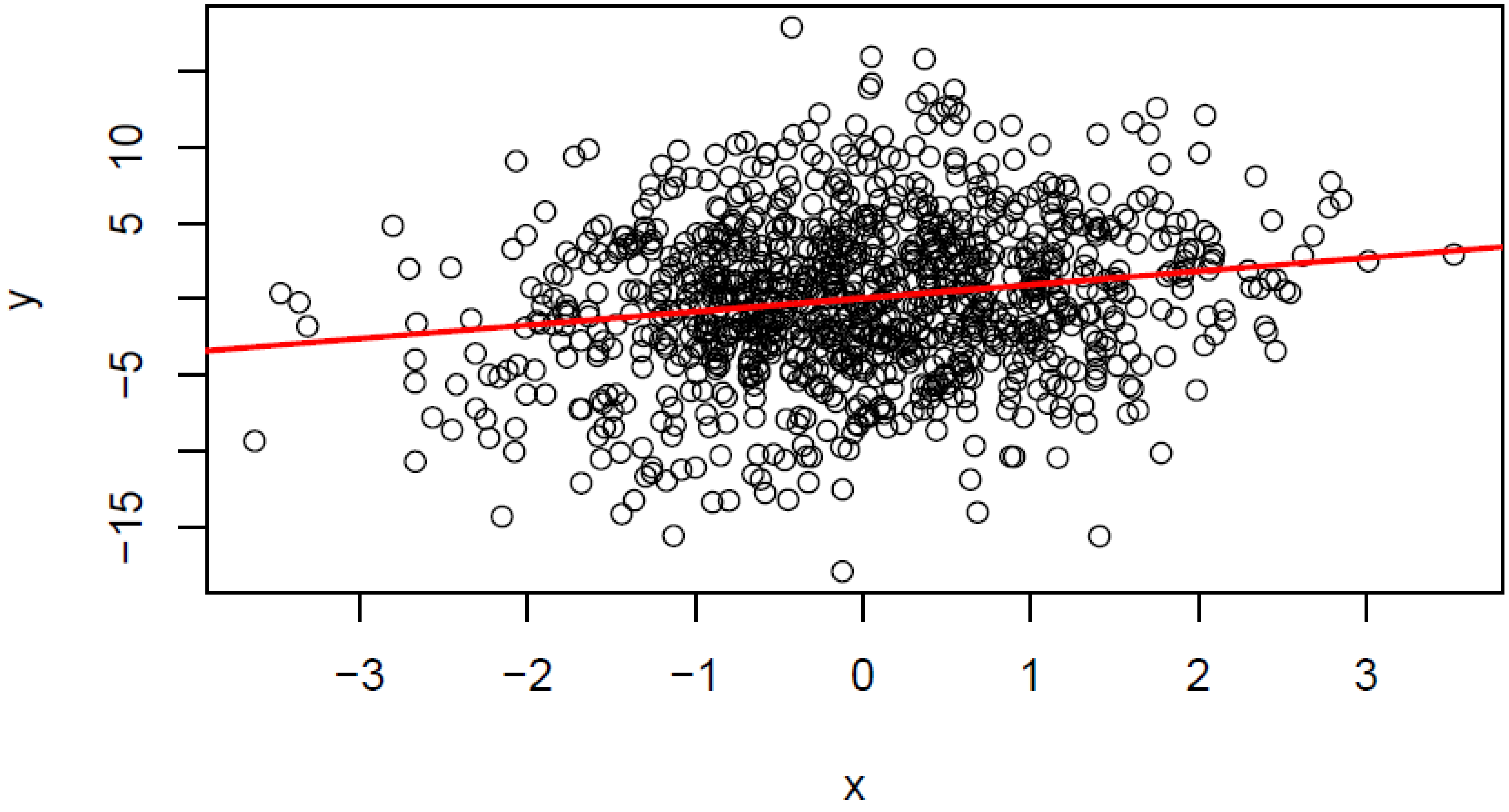


```r
u = rnorm(n, 0, 0.5);
y = u + x;
plot(x,y);
fit = lm(y ~ x);
summary(fit);
# Regression Line
abline(fit$coefficients, col = "red", lwd = 2);
```




```
u = rnorm(n, 0, 0.5);
y = u + x;
plot(x,y);
fit = lm(y ~ x);
summary(fit);
# Regression Line
abline(fit$coefficients, col = "red", lwd = 2);
```

```r
u = rnorm(n, 0, 5);
y = u + x;
plot(x,y);
fit = lm(y ~ x);
summary(fit);
# Regression Line
abline(fit$coefficients, col = "red", lwd = 2);
```




```
u = rnorm(n, 0, 5);
y = u + x;
plot(x,y);
fit = lm(y ~ x);
summary(fit);
# Regression Line
abline(fit$coefficients, col = "red", lwd = 2);
```

Fixed Effects and Clustering (lfe package)

- This package is currently under repair. So we need to download an earlier version
  - rtools provides a set of tools to build and install earlier version of softwares
  - `remotes::install_version("lfe");`
    - OR try: `remotes::install_github("cran/lfe");`
  - The above will take some time

Fixed Effects and Clustering (lfe package)

- This package is currently under repair. So we need to download an earlier version
  - rtools provides a set of tools to build and install earlier version of softwares
  - `remotes::install_version("lfe");`
    - OR try: `remotes::install_github("cran/lfe");`
  - The above will take some time
- We ran basic regression as: `lm(y ~ x)`
- Suppose we wish to add fixed effects, we can do
  - `lm(y ~ x + factor(fe_1) + factor(fe_2) + ...)`
  - The above works fine for a small number of fixed effects (FE)
  - `lm()` is prohibitively slow for large number of FE and FE with large num of levels

- There are two ways to take care of FE
  - Include them in regression
  - De-mean the variables (both x and y) with respect to those FE levels
    - `lfe:felm()` is very efficient at this

- There are two ways to take care of FE
  - Include them in regression
  - De-mean the variables (both x and y) with respect to those FE levels
    - `lfe:felm()` is very efficient at this
- `lm()` offers no support for clustering of standard errors
  - Most recent research includes some form of clustering in the results
    - In panel data, you often need multi-way clustering
    - Earlier approach was to correct for heteroscedasticity and autocorrelation (HAC)
      - You might recognize terms like White (Robust) standard errors, Newey-West adjustment, etc while reading papers
  - `lfe::felm()` provides inbuilt cluster robust standard errors

- There are two ways to take care of FE
  - Include them in regression
  - De-mean the variables (both x and y) with respect to those FE levels
    - `lfe:felm()` is very efficient at this
- `lm()` offers no support for clustering of standard errors
  - Most recent research includes some form of clustering in the results
    - In panel data, you often need multi-way clustering
    - Earlier approach was to correct for heteroscedasticity and autocorrelation (HAC)
      - You might recognize terms like White (Robust) standard errors, Newey-West adjustment, etc while reading papers
  - `lfe::felm()` provides inbuilt cluster robust standard errors
- Syntax: `felm(formula, data)`
  - formula: <MODEL> | <FE> | <INSTR> | <CLUSTERS>

```
felm(   y ~ x1 + x2   | f1 + f2 |   0   |   c1 + c2     )
```

?felm (Help page of felm command)

?felm (Help page of felm command)

- The formula specification is a response variable followed by a four part formula.
  - The first part consists of ordinary covariates, the second part consists of factors to be projected out. The third part is an IV-specification. The fourth part is a cluster specification for the standard errors.
  - I.e. something like `y ~ x1 + x2 | f1 + f2 | (Q|W ~ x3+x4) | clu1 + clu2` where `y` is the response, `x1,x2` are ordinary covariates, `f1,f2` are factors to be projected out, `Q` and `W` are covariates which are instrumented by `x3` and `x4`, and `clu1,clu2` are factors to be used for computing cluster robust standard errors.
  - Parts that are not used should be specified as `0`, except if it's at the end of the formula, where they can be omitted.
    - The parentheses are needed in the third part since | has higher precedence than ~.
  - Multiple left hand sides like `y|w|x ~ x1 + x2 |f1+f2|...` are allowed.

Different Regressions in R

Different Regressions in R

- OLS is very simple
  - `fit = lm(y ~ x)`
    - You can check the summary using `summary(fit)`. This gives std errors, t-stats and R2
  - `lfe::felm(y ~ x)`

Different Regressions in R

- OLS is very simple
  - `fit = lm(y ~ x)`
    - You can check the summary using `summary(fit)`. This gives std errors, t-stats and R2
  - `lfe::felm(y ~ x)`
- GLS
  - `nlme::gls(y ~ x, correlation = C, weights = w)`
    - C is the group correlation mx, while w are heteroscedasticity weights

Different Regressions in R

- OLS is very simple
  - `fit = lm(y ~ x)`
    - You can check the summary using `summary(fit)`. This gives std errors, t-stats and R2
  - `lfe::felm(y ~ x)`
- GLS
  - `nlme::gls(y ~ x, correlation = C, weights = w)`
    - C is the group correlation mx, while w are heteroscedasticity weights
- IV
  - We can use `lfe::felm` for IV regression
    - `lfe::felm(y ~ x1 + x2 | 0 | (x3|x4 ~ w3 + w4) | 0)`
    - `AER::ivreg(y ~ x1 + x2 + x3 + x4 | x1 + x2 + w3 + w4)`
  - Requirements:
    - Instruments (w3, w4) must be correlated with endogenous variables (x3, x4)
    - Instruments (w3, w4) are unrelated to the error term, i.e. instruments affect y only through endogenous variables x3 and x4

- Logit and Probit Models
    - `glm(y ~ x1 + x2, family = binomial("logit"))`
    - `glm(y ~ x1 + x2, family = binomial("probit"))`
    - Try `?family` for more insights into how to use other models

- Logit and Probit Models
  - `glm(y ~ x1 + x2, family = binomial("logit"))`
  - `glm(y ~ x1 + x2, family = binomial("probit"))`
  - Try `?family` for more insights into how to use other models
- For more details check out `VGAM::vglm` which implements vectorised glm for several other models
- The need to perform a regression other than plain vanilla OLS rarely arises
  - Important exceptions are IV (2-SLS), FE and clustering.
  - Refer “Introductory Econometrics” by Woolridge for more theory and details on different regression models.

- Logit and Probit Models
  - `glm(y ~ x1 + x2, family = binomial("logit"))`
  - `glm(y ~ x1 + x2, family = binomial("probit"))`
  - Try `?family` for more insights into how to use other models
- For more details check out `VGAM::vglm` which implements vectorised glm for several other models
- The need to perform a regression other than plain vanilla OLS rarely arises
  - Important exceptions are IV (2-SLS), FE and clustering.
  - Refer “Introductory Econometrics” by Woolridge for more theory and details on different regression models.
- Bootstrapping
  - In some models, its impossible to accurately judge the structure of standard errors. There we can employ boot-strapping to get standard errors.
    - Sample N data points (with repetition) from your dataset and estimate the model
      - Do this 1000 (or more) times
      - The standard error of coef. errors is then your standard errors

Time-series Regressions

- The time-series regression is specified the same way as a cross-sectional regression

Time-series Regressions

- The time-series regression is specified the same way as a cross-sectional regression
- But we need to be careful about some issues
  - Non-stationarity in data
    - 1st/2nd order integration?
  - Persistent time-series
    - AR (autoregressive) and moving average (MA) parameters
  - Fit `arima(x, order)` model where `order = c(p, d, q)`
    - `p`: AR order, `d`: integration order, `q`: MA order
  - Spurious Regression
    - Will return of SBI predict return of HDFC? Reliance?
      - Nifty (or broader market) return predicts both SBI and Reliance!
  - Co-integration
    - Does India's GDP forecasts predict Nifty levels?
- Learn more at https://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model