Cheat Sheet: Linear Regression

Measurement and Evaluation of HCC Systems

Scenario

Use regression if you want to test the simultaneous linear effect of several variables `varX1`, `varX2`, … on a continuous outcome variable `varY`. In this scenario, you are predicting `varY` with `varX1`, `varX2`, ….

Power analysis for linear regression

- There are several versions of power analysis for linear regression, but the most versatile one is "*F* tests", "Linear multiple regression: Fixed model, R^2 increase".
- A power analysis has four variables: Effect size, α (usually .05), power (usually .85), and *N*. If you know three of these, G*Power will calculate the fourth. Select the correct type of power analysis, based on the information you have, and what you want to find out.
- "Number of tested predictors" is the number of *X*s that you want to test simultaneously. Often you want to test the effect of an individual *X*, in which case this field should say 1.
- "Total number of predictors" is the total number of *X*s in the model.
- By clicking on "Determine", you can compute the effect size f^2 from the partial R^2. This is the increase in R^2 caused by the predictors you want to test.
- Click on "Calculate" to calculate the missing parameter.

Plotting scatterplots with linear trend line

- Use the `ggplot2` package to plot a scatterplot with a linear trend line.
  ```
  ggplot(data, aes(varX1, varY)) + geom_point() + geom_smooth(method="lm", color="red", se=F)
  ```
- (optional) To add a mean line, add:
  ```
  + geom_line(aes(y = mean(data$varY)), color="blue"
  ```
- Visually inspect if the relationship is indeed a linear one (see assumptions). You can also check if the scatterplot has a funnel shape, which may suggest heteroscedasticity (see assumptions). Repeat this procedure for `varX2`, etc.

Pre-testing assumptions

- In a linear regression, *Y* should be independent, continuous, and unbounded. The error variance should be normally distributed, which is true if *Y* is normally distributed.
- If your *N* is small:
  - Test for significant skewness, kurtosis, and Shapiro-Wilk test using `stat.desc` in the `pastecs` package.
    ```
    stat.desc(data$varY, basic=F, norm=T)
    ```
  - Multiply `skew.2SE` and `kurt.2SE` by 2 to get the *Z*-scores of skewness and kurtosis. Compare these values to typical cut-off values ($Z > \pm 1.96$: $p < .05$, $Z > \pm 2.58$: $p < .01$, $Z > \pm 3.29$: $p < .001$). The significance of the Shapiro-Wilk test is listed under `normtest.p`.
- If your *N* is large:
  - Draw the histogram for `varY`, overlaid with normal curves (using `ggplot2`), and visually inspect whether they follow the normal distribution:
    ```
    ggplot(data, aes(varY)) + geom_histogram(aes(y=..density..), binwidth=1, color="black", fill="white") + stat_function(fun = dnorm, args = list(mean = mean(data$varY), sd = sd(data$varY)))
    ```
  - Change the `binwidth` setting based on what is suitable for your data.
  - Draw normal a Q-Q plot, and visually inspect whether the data follows the diagonal line:
    ```
    qplot(sample = data$varY, stat="qq")
    ```
- If you have dummy variables (see below), you want to test normality within each group (see the *t* test and ANOVA cheat sheets).
- If your data has positive skew, and your data only has positive values, you can possibly fix this by transforming your *Y* variable:
  - Log transform:
    ```
    data$varYlog <- log(data$varY + 1)
    ```
  - Or, square root transform:
    ```
    data$varYsqrt <- sqrt(data$varY)
    ```
- In other cases of violations of assumptions, you can conduct a robust test (see below).
- Aside from normality, you also need to test for linearity, homoscedasticity and a lack of multicollinearity. You can do this after the test using the residual plots of the model.

(optional) Preparing dummy variables

- If one or more of your *X*s are nominal variables, you need to create dummy variables for them. R does this automatically, as long as the nominal *X*s are coded as factors. If they are coded as numerical values (e.g. 1, 2, 3…), you can recode them as factors using the `revalue`

function in the `plyr` package:
`data$varX1 <- revalue(as.factor(data$varX1), c("1"="cat1", "2"="cat2"))`

- `cat1` and `cat2` are descriptive names of the different categories of the variable. You can revalue more categories if needed.
- R automatically selects a baseline category against which all other categories are compared. If it chooses this category incorrectly, you can `relevel` the variable, e.g., to make `cat1` the baseline category:
  `data$varX1 <- relevel(data$varX1, ref="cat1")`
- Note that when you have a nominal variable with more than two categories, the automatically created dummies form a non-orthogonal contrast. This means that you should apply a correction (e.g. Bonferroni, Holm, or Benjamini-Hochberg) to the *p*-values of these contrasts. Alternatively, you can create orthogonal contrasts for such variables. Both are discussed in the ANOVA cheat sheet.

Running the test

- Run the regression model as follows (include additional *X*s if needed):
  `model1 <- lm(varY ~ varX1 + varX2, data = data)`
- Get the model summary:
  `summary(model1)`
- In terms of overall model fit, this output will give you the Multiple R-squared, which is the proportion of the variance of `varY` explained by the model. The Adjusted R-squared is the expected R^2 if the test were to be repeated.
- The *F*-statistic and its *p*-value tells us whether the model makes a significantly better prediction than the grand mean.
- Each coefficient represents the effect of an *X* on *Y*, given all the other *X*s. If *X* increases/decreases by 1, *Y* is expected to increase/decrease by this amount. For dummy variables, the coefficient represents the difference in *Y* between this category and the baseline category.
- Each coefficient has a *t* test and a *p*-value to test if the effect is significant. Divide the *p*-value by 2 if you were conducting a one-sided test (i.e. if you had a directional hypothesis). You can get the effect size *r* using the formula $r = \sqrt{(t^2 / t^2 + df)}$.
- You can get confidence intervals for the coefficients using the `confint` function:
  `confint(model1)`
- To compare the effects of different Xs, you need to standardize them using `lm.beta` in the `QuantPsyc` package:
  `lm.beta(model1)`
  This gives you the beta coefficients. If *X* increases/decreases by 1 standard deviation, *Y* is

expected to increase/decrease by this proportion of its standard deviation. Note: this does not work for dummy variables.

(optional) Robust versions

- You can bootstrap the regression coefficients (the *b*s) of a linear regression. First create a function for running the bootstrap sample (include additional *X*s if needed):

```
bootFun <- function(sample,i){
    fit <- lm(varY~varX1+varX2, data=sample[i,])
    return(coef(fit))
}
```

- Then run the bootstrap sample over the function 2000 times:

```
bootResult <- boot(data, bootFun, 2000)
```

- Get the output; the `original` column shows the regression coefficient in the original sample, the `bias` column shows the difference between this and the coefficient in the bootstrap sample, and the `std. error` column shows the bootstrapped standard error. Row `t1*` shows the intercept, the subsequent rows show the coefficient *b* for each *X*:

```
bootResult
```

- Get the confidence interval of the first coefficient; the `BCa` version is the most robust variant:

```
boot.ci(bootResult,index=2)
```

- Repeat with `index=3` etc. for the other coefficients.

(optional) Testing additional variables

- In any regression, always first decide on your outcome variable (*Y*) and the most important *X*s. For additional *X*s, only include them if they correlate with *Y*. Do not include *X*s that are correlated too highly with any of the other *X*s ($r > .8$ or $r < –.8$), because this may lead to multicollinearity (see below).
- If these additional *X*s have a clear hierarchy (some are more interesting than others), then add them step-by-step in order of importance. Otherwise, add them all at once and remove non-significant ones one-by-one.
- Models with more Xs always have a higher R^2. You can test whether this increase in R^2 is significant using an *F*-ratio test:

```
anova(model1,model2)
```

- Do not forget to assess the *b* estimates of the new model. Even the *b* estimates for the old parameters may have changed, especially when the new *X*s are correlated with the old *X*s.

Post-testing assumptions

- Run the plots of the model:

```
plot(model1)
```

- The first plot shows the residuals ε at different levels of *Y*. If this plot is funnel-shaped, you have heteroscedasticity. If the red mean line is not a straight line, you have non-linearity.
- The next plot shows the normal Q-Q plot of the residuals. If they deviate strongly from the diagonal line, you have non-normality.
- With any of these problems, you can possibly fix this by transforming your *Y* variable (see "Pre-testing assumptions"), or by running a bootstrapped regression (see "Robust versions").
- Test for multicollinearity of your *X*s using the Variance Inflation Factor (VIF); these should be below 5:
  ```
  vif(model1)
  ```
- Check the average VIF. Ideally this should be 1, but values close to (say, up to 1.5) are permissible:
  ```
  mean(vif(model1))
  ```
- If you find any problems, some of your Xs are probably highly correlated with each other. Remove one of these Xs and start over.

Inspecting outliers

- Save the standardized residuals to your data:
  ```
  data$rstand <- rstandard(model1)
  ```
- Create a filter for large standardized residuals:
  ```
  data$rstand.large <- (data$rstand > 1.96 | data$rstand < -1.96)
  ```
- Inspect these outliers:
  ```
  data[data$rstand.large,]
  ```
- If these outliers comprise more than 5% of the data, you have too many data for which the model does not fit very well. Moreover, if more than 1% of the data has an `rstand` > 2.58, you have too many data for which the model is a very poor fit. Finally, any data with an `rstand` > 3.29 are extreme outliers: they will likely have to be removed.
- Save Cook's distances, hat values (leverage), and covariance ratios to your data:
  ```
  data$cook <- cooks.distance(model1)
  data$leverage <- hatvalues(model1)
  data$covratio <- covratio(model1)
  ```
- Check out the Cook's distances, hat values, and covariance ratio for the data with a large residual:
  ```
  data[data$rstand.large,c("cook", "leverage", "covratio")]
  ```
- Cook's distance should be smaller than 1, leverage should be smaller than 3*(*k*+1)/*N*), and the covariance ratio should be between 1-(3(*k*+1)/*N*) and 1+(3(*k*+1)/*N*), where *k* is the number of *X*s, and *N* is the sample size.
- These metrics are a bit sensitive to sample size, so for large samples, focus on the Cook's distance. In case of problems, you can possibly fix this by removing the offending data point,

transforming your *Y* variable (see "Pre-testing assumptions"), or running a bootstrapped regression (see "Robust versions").

Reporting

- Create a table to report your regression model(s); in the example below, model 1 has two continuous *X*s, and model 2 adds a continuous *X* and a nominal *X* with 3 categories:

	ΔR^2	*B*	*SE B*	β	*P*
Model 1	[R^2]				[*p*-value of model *F*-test]
Constant		[estimate]	[std. error]		
[varX1]		[estimate]	[std. error]	[beta coef.]	[*p*-value]
[varX2]		[estimate]	[std. error]	[beta coef.]	[*p*-value]
Model 2	[R^2 increase]				[*p*-value of *F*-ratio test between model 1 and 2]
Constant		[estimate]	[std. error]		
[varX1]		[estimate]	[std. error]	[beta coef.]	[*p*-value]
[varX2]		[estimate]	[std. error]	[beta coef.]	[*p*-value]
[varX3]		[estimate]	[std. error]	[beta coef.]	[*p*-value]
[varX4] baseline: [cat1] [cat2] [cat3]		 [estimate] [estimate]	 [std. error] [std. error]		 [*p*-value] [*p*-value]

- You can additionally highlight and explain individual effects:
  - "Controlling for [varX2], [varX3], and [varX4], each 1 point increase in [varX1] is related to [estimate] points increase in [varY], *t*([deg. freedom]) = .xx, *p* = .xxx."
  - "Controlling for [varX1], [varX2], and [varX3], participants with [cat2] had a [estimate] points higher [varY] than participants with [cat1], *t*([deg. freedom]) = .xx, *p* = .xxx."