Introduction to Statistics in R Presented by:

BROWN

Center for

Biomedical Informatics

BROWN
Center for
Biomedical Informatics

Introduction to Statistics in R

Day 7 - Non-Parametric Statistics in R

Adam J Sullivan

Non Parametric Statistics

What are they?

- Non parametrics means that you do not need to specify a specific distribution for the data.
- Many of the methods you have learned up to this point require data dealing with the normal distribution.
- This is due partially to the fact that the t-distribution, χ^2 distribution, and the F distribution can all be derived from the normal distribution.
- Traditionally you are just taught to use normallity and made to either transform the data or just go ahead knowing it is incorrect.

Normal vs Skewed Data

- Data that is normally distributed has:
  - the mean and the median the same.
  - The data is centered about the mean.
  - Very specific probability values.
- Data that is skewed has:
  - Mean less than median for left skewed data.
  - Mean greater than median for right skewed data.

Why is this an issue?

- In 1998 a survey was given to Harvard students who entered in 1973:
    - The mean salary was $750,000
    - The median salary was $175,000
- What could be a problem with this?
- What happened here?

Why do we use Parametric Models?

1. Parametric Models have more power so can more easily detect significant differences.
2. Given large sample size Parametric models perform well even in non-normal data.
3. Central limit theorem states that in research that can be perfromed over and over again, that the means are normally distributed.
4. There are methods to deal with incorrect variances.

Why do we use Non Parametric Models?

1. Your data is better represented by the median.
2. You have small sample sizes.
3. You cannot see the ability to replicate this work.
4. You have ordinal data, ranked data, or outliers that you can’t remove.

What Non Parametric Tests will we cover?

- Sign Test
- Wilcoxon Signed-Rank Test
- Wilcoxon Rank-Sum Test (Mann-Whitney U Test, ...)
- Kruskal Wallis test
- Spearman Rank Correlation Coefficient
- Bootstrapping

The sign Test

The Sign Test

- The sign test can be used when comparing 2 samples of observarions when there is not independence of samples.
- It actually does compares matches together in order to accomplish its task.
- This is similar to the paired t-test
- No need for the assumption of normality.
- Uses the Binomial Distribution

Steps of the Sign Test

- We first match the data
- Then we subtract the 2nd value from the 1st value.
- You then look at the sign of each subtraction.
- If there is no difference between the two groups you shoul have roughly 50% positives and 50% negatives.
- Compare the proportion of positives you have to a binomial with p=0.5.

Example: Binomial Test Function

- Consider the scenario where you have patients with Cystic Fibrosis and health individuals.
- Each subject with CF has been matched to a healthy individual on age, sex, height and weight.
- We will compare the Resting Energy Expenditure (kcal/day)

Reading in the Data

```
library(readr)
ree <- read_csv("ree.csv")
ree
```

```
## # A tibble: 13 x 2
##       CF Healthy
##    <int>   <int>
##  1  1153     996
##  2  1132    1080
##  3  1165    1182
##  4  1460    1452
##  5  1634    1162
##  6  1493    1619
##  7  1358    1140
##  8  1453    1123
##  9  1185    1113
## 10  1824    1463
## 11  1793    1632
## 12  1930    1614
## 13  2075    1836
```

Function in R

- Comes from the `BDSA` Package

```
SIGN.test(x ,y, md=0, alternative = "two.sidesd",
conf.level=0.95)
```

- Where
  - `x` is a vector of values
  - `y` is an optional vector of values.
  - `md` is median and defaults to 0.
  - `alternative` is way to specific type of test.
  - `conf.level` specifies $1 - \alpha$.

Our Data

```
library(BSDA)
attach(ree)
SIGN.test(CF, Healthy)
detach()
```

```
##
##  Dependent-samples Sign-Test
##
## data:  CF and Healthy
## S = 11, p-value = 0.02246
## alternative hypothesis: true median difference is not equal to 0
## 95 percent confidence interval:
##   25.35385 324.47832
## sample estimates:
## median of x-y
##           161
##
## Achieved and Interpolated Confidence Intervals:
##
##                   Conf.Level  L.E.pt   U.E.pt
## Lower Achieved CI     0.9077 52.0000 316.0000
## Interpolated CI       0.9500 25.3538 324.4783
## Upper Achieved CI     0.9775  8.0000 330.0000
```

By "Hand"

- Subtract Values

```
library(tidyverse)
ree <- ree %>%
mutate(diff = CF - Healthy)
ree
```

```
## # A tibble: 13 x 3
##        CF Healthy  diff
##     <int>   <int> <int>
##  1  1153     996   157
##  2  1132    1080    52
##  3  1165    1182  - 17
##  4  1460    1452     8
##  5  1634    1162   472
##  6  1493    1619  -126
##  7  1358    1140   218
##  8  1453    1123   330
##  9  1185    1113    72
## 10  1824    1463   361
## 11  1793    1632   161
## 12  1930    1614   316
## 13  2075    1836   239
```

- count negatives

By "Hand"

```
binom.test(2,13)
```

```
##
##  Exact binomial test
##
## data:  2 and 13
## number of successes = 2, number of trials = 13, p-value = 0.02246
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.01920667 0.45447106
## sample estimates:
## probability of success
##              0.1538462
```

Wilcoxon Signed-Rank Test

Wilcoxon Signed-Rank Test

- The sign test works well but it truly ignores the magntiude of the differences.
- Sign test often not used due to this problem.
- Wilcoxon Signed Rank takes into account both the sign and the rank

How does it work?

- Pairs the data based on study design.
- Subtracts data just like the sign test.
- Ranks the magnitude of the difference:

```
8
-17
52
-76
```

What happens with these ranks?

SUBTRACTION	POSITIVE RANKS	NEGATIVE RANKS
8	1	
-17		-2
52	3	
-76		-4
Sum	4	**-6**

What about after the sum?

- $W_+ = 1 + 3 = 4$
- $W_- + -2 + -4 = -6$
- Mean: $\frac{n(n+1)}{4}$
- Varince: $\frac{n(n+1)(2n+1)}{24}$
- Any ties, t: decrease variance by $t^3 - \frac{t}{48}$
- z test:

$$z = \frac{W_{smaller} - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24} - t^3 - \frac{t}{48}}}$$

Wilcoxon Signed Rank in R

```
attach(ree)
wilcox.test(CF, Healthy, paired=T)
```

```
##
##  Wilcoxon signed rank test
##
## data:  CF and Healthy
## V = 84, p-value = 0.004639
## alternative hypothesis: true location shift is not equal to 0
```

Wilcoxon Rank-Sum Test

Wilcoxon Rank-Sum Test

- This test is used on indepdent data.
- It is the non-parametric version of the 2-sample t-test.
- Does not requre normality or equal variance.

How do we do it?

- Order each sample from least to greatest
- Rank them.
- Sum the ranks of each sample

What do we do with summed ranks?

- W_s smaller of 2 sums.
- Mean: $\frac{n_s(n_s + n_L + 1)}{2}$
- Variance: $\frac{n_s n_L(n_s + n_L + 1)}{12}$
- z-test

$$z = \frac{W_s - \frac{n_s(n_s + n_L + 1)}{2}}{\sqrt{\frac{n_s n_L(n_s + n_L + 1)}{12}}}$$

Wilcoxon Rank-Sum in R

- Consider built in data `mtcars`

```
library(tidyverse)
cars <- as_data_frame(mtcars)
cars
```

```
## # A tibble: 32 x 11
##      mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
##  * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  21.0  6.00   160 110    3.90  2.62  16.5  0     1.00  4.00  4.00
##  2  21.0  6.00   160 110    3.90  2.88  17.0  0     1.00  4.00  4.00
##  3  22.8  4.00   108  93.0  3.85  2.32  18.6  1.00  1.00  4.00  1.00
##  4  21.4  6.00   258 110    3.08  3.22  19.4  1.00  0     3.00  1.00
##  5  18.7  8.00   360 175    3.15  3.44  17.0  0     0     3.00  2.00
##  6  18.1  6.00   225 105    2.76  3.46  20.2  1.00  0     3.00  1.00
##  7  14.3  8.00   360 245    3.21  3.57  15.8  0     0     3.00  4.00
##  8  24.4  4.00   147  62.0  3.69  3.19  20.0  1.00  0     4.00  2.00
##  9  22.8  4.00   141  95.0  3.92  3.15  22.9  1.00  0     4.00  2.00
## 10  19.2  6.00   168 123    3.92  3.44  18.3  1.00  0     4.00  4.00
## # ... with 22 more rows
```

Wilcoxon Rank-Sum in R

- We will Consider `mpg` and `am`
- `mpg`: Miles Per Gallon on Average
- `am`
  - 0: automatic transmission
  - 1: manual transmission

Wilcoxon Rank-Sum in R

```
attach(cars)
wilcox.test(mpg, am)
detach(cars)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  mpg and am
## W = 1024, p-value = 2.75e-12
## alternative hypothesis: true location shift is not equal to 0
```

Kruskal Wallis Test

Kruskal Wallis Test

- If we have multiple groups of independent data that are not normally distributed or have variance issues, you can use the Kruskal Wallis Test.
- It tests significant differences in medians of the groups.
- This is a non-parametric method for One-Way ANOVA.
- Harder to try and calculate by hand, so we will just use R.

Kruskal Wallis Test in R

```
kruskal.test(formula, data, subset, ...)
```

- Where
  - `formula` is `y~x` or can enter `outcome,group` instead.
  - `data` is the dataframe of interest.
  - `subset` if you wish to test subset of data.

Arthritis Data

- comes from the `BSDA` package.
- `Arthriti`

VARIABLE	DESCRIPTION
`time`	Time in Days until patient felt relief
`treatment`	Factor with three levels `A`, `B`, and `C`

Arthritis Data

```r
library(BSDA)
Arthriti
```

```
## # A tibble: 51 x 2
##     time treatment
##    <int> <fctr>
##  1    40 A
##  2    35 A
##  3    47 A
##  4    52 A
##  5    31 A
##  6    61 A
##  7    92 A
##  8    46 A
##  9    50 A
## 10    49 A
## # ... with 41 more rows
```

Kruskal-Wallis Test in R

```
kruskal.test(time~treatment, data=Arthriti)
```

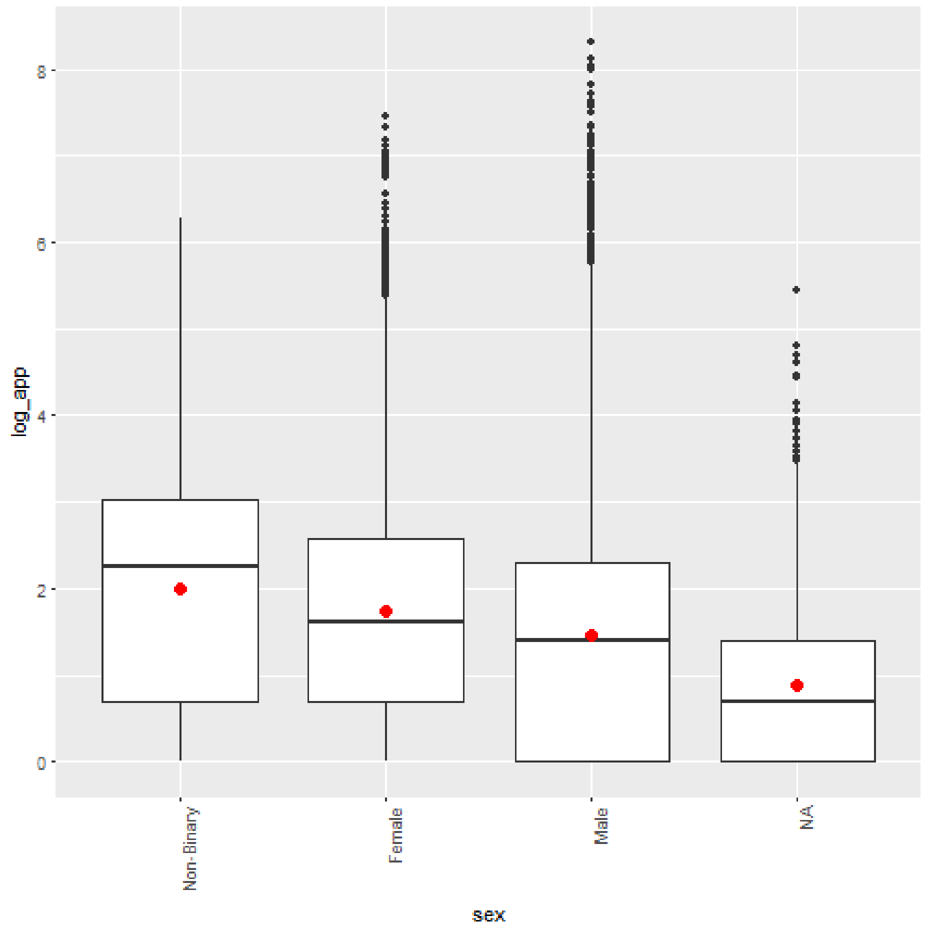
```
##
##  Kruskal-Wallis rank sum test
##
## data:  time by treatment
## Kruskal-Wallis chi-squared = 1.8335, df = 2, p-value = 0.3998
```

Spearman Rank Correlation Coefficient

Spearman Rank Correlation Coefficient

- Correlation is a measurement of the strength of a linear relationship between variables.
- This means it does not necessarily get the actual magnitude of relationship.
- Spearman Rank Correlation seeks to fix this.
- It works with Montonic Data.

Monotonic Data



Spearman Rank Correlation in R

- We can do this the the `cor()` function.

```
#Pearson from Monotonic Decreasing
cor(x2,y2, method="pearson")

#Spearman from Monotonic Decreasing
cor(x2,y2, method="spearman")
```

```
## [1] -0.888548
## [1] -0.9962037
```

Bootstrapping

Bootstrapping

- Linear regression itself works well even with non-normal errors.
- What can happen with non-normal errors is that our standard error is estimated incorrectly and therefore we cannot trust our p-values or confidence intervals.
- Bootstrap is a non-parametric technique to get an idea of variance.

Bootstrapping

- Treats your data as a population.
- Draws samples from the data with replacement.
- Steps:
  1. So if you have 100 subjects, you draw a sample of size 100 from all 100 subjects, some may get picked twice or more and others not at all.
  2. Then run your regression over this dataset.
  3. Pull out coefficients.
  4. Repeat steps 1-3 say 1000 times.
  5. Get 95% interval from now normal data.

Bootstrapping in R

- Use the `boot` package.

```
boot(data, statistic, R, ....)
```

- Where
  - `data` is your dataframe of interest.
  - `statistic` is what you want to bootstrap
  - `R` how many replicates do you want.
  - `...` other options from `statistic` or `boot` function.

Bootstrapping Regression Coefficients

```
# Bootstrap 95% CI for regression coefficients
library(boot)
# function to obtain regression weights
bs <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(coef(fit))
}
# bootstrapping with 1000 replications
results <- boot(data=mtcars, statistic=bs,
    R=1000, formula=mpg~wt+disp)

# view results
results
plot(results, index=1) # intercept
plot(results, index=2) # wt
plot(results, index=3) # disp

# get 95% confidence intervals
boot.ci(results, type="bca", index=1) # intercept
boot.ci(results, type="bca", index=2) # wt
boot.ci(results, type="bca", index=3) # disp
```

Bootstraping R^2

```
# Bootstrap 95% CI for R-Squared
library(boot)
# function to obtain R-Squared from the data
rsq <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=d)
  return(summary(fit)$r.square)
}
# bootstrapping with 1000 replications
results <- boot(data=mtcars, statistic=rsq,
    R=1000, formula=mpg~wt+disp)

# view results
results
plot(results)

# get 95% confidence interval
boot.ci(results, type="bca")
```

Other Methods

Other Methods

- There are various other Methods that we do not have the ability to really discuss here
  - Generalized Additive Models
  - Splines and Other penalized Regressions
  - Classification and Regression Trees
  - Smoothing Regressions
  - Permutation Tests