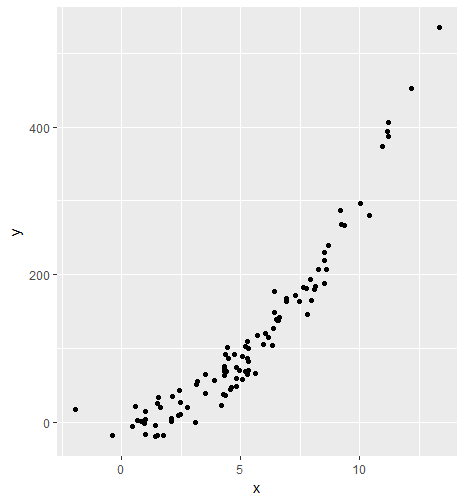
**Plotting the best fitting line, Plotting points along the least square line, Visualizing the Residuals,**

**Best fitting line for polynomial degree 2**

ggplot()+

geom\_point(data=dat, aes(x=x, y=y))



We will find the best fitting line using lm function

> lm(y~x, data=dat)

Call:

lm(formula = y ~ x, data = dat)

Coefficients:

(Intercept) x

-65.27 34.04

x1 <- function(x){

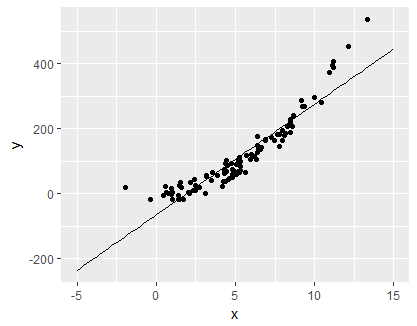
34.04\*x-65.27

}

ggplot()+

geom\_point(data=dat, aes(x=x, y=y))+

stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=x1)



**Plotting the points along least squared line**

f <- function(x){

34.04\*x-65.27

}

ggplot()+

geom\_point(data=dat, aes(x=x, y=y))+

stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=f)

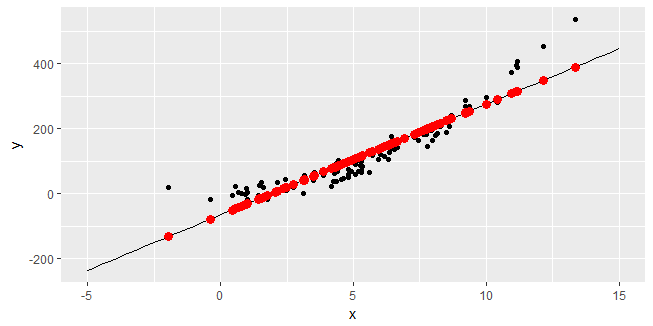
|  |
| --- |
| > x <- dat$x  > y <- f(x)  > means <- data.frame(x,y) |
|  |
| |  | | --- | | > | |

ggplot()+

geom\_point(data=dat, aes(x=x, y=y))+

stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=f)+

geom\_point(data=means, aes(x=x, y=y), color='red', size=3)



Now if you take a look at block point it will straight drop a red point on the line that corresponds to that data point.

**Visualizing the Residuals**

**How do we get a line from points to red dot on the line?**

|  |
| --- |
| > head(dat)  x y  1 7.2940069 172.106043  2 7.7541205 181.471842  3 0.4429789 -5.515109  4 4.9272564 69.539387  5 6.4762602 139.634188  6 4.2627953 37.708302 |
|  |
| |  | | --- | | > | |

We will create a third column here, which will hold the groups. As we have many red points on red line.

|  |
| --- |
| > dat$group <- 1:100  > head(dat)  x y group  1 7.2940069 172.106043 1  2 7.7541205 181.471842 2  3 0.4429789 -5.515109 3  4 4.9272564 69.539387 4  5 6.4762602 139.634188 5  6 4.2627953 37.708302 6 |
|  |
| |  | | --- | | > | |

We will also create this third column group in means dataframe

|  |
| --- |
| > means$group <- 1:100  > head(means)  x y group  1 7.2940069 183.01800 1  2 7.7541205 198.68026 2  3 0.4429789 -50.19100 3  4 4.9272564 102.45381 4  5 6.4762602 155.18190 5  6 4.2627953 79.83555 6 |
|  |
| |  | | --- | |  | |

We will bind these both data frames using row bind function in r

|  |
| --- |
| > groups <- rbind(dat, means)  > head(groups)  x y group  1 7.2940069 172.106043 1  2 7.7541205 181.471842 2  3 0.4429789 -5.515109 3  4 4.9272564 69.539387 4  5 6.4762602 139.634188 5  6 4.2627953 37.708302 6 |
|  |
| |  | | --- | | > | |

Lets call groups, see what we got. That will give you an idea what rbind does.

|  |
| --- |
| x y group  1 7.2940069 172.1060430 1  2 7.7541205 181.4718418 2  3 0.4429789 -5.5151093 3  4 4.9272564 69.5393874 4  5 6.4762602 139.6341885 5  6 4.2627953 37.7083019 6  7 1.4814663 -17.5066462 7  8 5.7172175 117.7765915 8  9 7.9703463 164.9854584 9  10 5.6110566 65.7417582 10  11 4.8018845 59.9750286 11  12 9.2016230 287.5074752 12  13 3.4964242 39.4662091 13  14 8.4986622 188.5323612 14  15 2.4390091 43.0218809 15  16 4.3268831 75.1872954 16  17 0.6634603 2.8395208 17  18 3.5220913 64.4865112 18  19 7.7905918 145.5677092 19  20 4.8224992 73.5766385 20  21 5.2013939 103.1572737 21  22 12.1501566 452.0065528 22  23 5.0855737 58.0930091 23  24 8.0774541 180.0513012 24  25 5.2076196 69.0727932 25  26 3.1734968 55.9252764 26  27 9.3565758 267.2404402 27  28 0.9396314 -1.4247625 28  29 8.6941338 239.2231695 29  30 6.0277782 120.7310201 30  31 4.7309894 91.8504012 31  32 10.4130016 280.5152627 32  33 3.8911173 57.1809274 33  34 4.3501971 91.9950749 34  35 4.4514737 101.9234510 35  36 4.6128661 47.0867698 36  37 0.8049878 1.2354100 37  38 8.6041282 207.4248661 38  39 6.1734584 114.6131483 39  40 9.2143414 267.6538444 40  41 7.9396499 193.6584081 41  42 8.5006067 219.2648972 42  43 1.7519184 -17.5505760 43  44 10.0053841 296.1560045 44  45 5.2656108 86.7749649 45  46 2.1412162 34.4548952 46  47 2.7570764 20.5026712 47  48 11.1854986 406.9095452 48  49 1.4238547 -19.1435871 49  50 8.2682993 206.8012982 50  51 6.4014518 148.2028122 51  52 5.3088876 70.2718456 52  53 1.6174395 19.9555354 53  54 5.9485336 104.8239590 54  55 3.1546825 51.6223267 55  56 5.3005320 100.2820431 56  57 2.4789963 26.1246286 57  58 0.9767535 14.7624925 58  59 -1.9511170 17.7934038 59  60 6.9170995 167.1061475 60  61 5.2855369 64.2652527 61  62 4.4087823 69.0716570 62  63 3.1037112 0.2113012 63  64 1.5224966 33.9908170 64  65 4.4883154 87.0645835 65  66 2.4841338 10.6462375 66  67 7.4551766 163.4146897 67  68 5.0847232 88.9488967 68  69 11.1806196 387.2229086 69  70 2.0649303 4.4836213 70  71 6.3135271 104.1472931 71  72 2.1035187 0.8262258 72  73 1.0137172 3.7070945 73  74 4.3228148 73.4667830 74  75 7.6344148 182.6203629 75  76 4.3454526 36.4749786 76  77 6.6418251 141.7282040 77  78 1.4828424 25.1672354 78  79 8.4935757 229.6835400 79  80 13.3411820 534.8048089 80  81 6.9229680 163.5427080 81  82 4.1925156 22.3715309 82  83 -0.3873840 -17.9436287 83  84 1.0014070 -16.4523610 84  85 4.8278364 49.0711919 85  86 6.5873198 138.2831609 86  87 11.1557332 394.5476008 87  88 2.3726048 9.7009010 88  89 0.5684373 21.2806841 89  90 8.1500537 183.8829008 90  91 6.4236014 177.1954274 91  92 5.3376159 82.3234837 92  93 6.5534456 137.4284261 93  94 5.2696975 109.9293402 94  95 1.4119448 -5.0674503 95  96 4.5616554 44.5818042 96  97 10.9431095 373.5335268 97  98 6.3739809 126.7677347 98  99 4.3079963 68.4620740 99  100 4.2953062 63.0300222 100  101 7.2940069 183.0179962 1  102 7.7541205 198.6802622 2  103 0.4429789 -50.1909972 3  104 4.9272564 102.4538083 4  105 6.4762602 155.1818961 5  106 4.2627953 79.8355533 6  107 1.4814663 -14.8408859 7  108 5.7172175 129.3440847 8  109 7.9703463 206.0405876 9  110 5.6110566 125.7303673 10  111 4.8018845 98.1861472 11  112 9.2016230 247.9532480 12  113 3.4964242 53.7482795 13  114 8.4986622 224.0244620 14  115 2.4390091 17.7538702 15  116 4.3268831 82.0171017 16  117 0.6634603 -42.6858111 17  118 3.5220913 54.6219863 18  119 7.7905918 199.9217444 19  120 4.8224992 98.8878717 20  121 5.2013939 111.7854499 21  122 12.1501566 348.3213324 22  123 5.0855737 107.8429276 23  124 8.0774541 209.6865368 24  125 5.2076196 111.9973721 25  126 3.1734968 42.7558322 26  127 9.3565758 253.2278412 27  128 0.9396314 -33.2849455 28  129 8.6941338 230.6783153 29  130 6.0277782 139.9155716 30  131 4.7309894 95.7728788 31  132 10.4130016 289.1885728 32  133 3.8911173 67.1836338 33  134 4.3501971 82.8107110 34  135 4.4514737 86.2581661 35  136 4.6128661 91.7519612 36  137 0.8049878 -37.8682138 37  138 8.6041282 227.6145227 38  139 6.1734584 144.8745254 39  140 9.2143414 248.3861803 40  141 7.9396499 204.9956815 41  142 8.5006067 224.0906529 42  143 1.7519184 -5.6346978 43  144 10.0053841 275.3132737 44  145 5.2656108 113.9713900 45  146 2.1412162 7.6169983 46  147 2.7570764 28.5808793 47  148 11.1854986 315.4843710 48  149 1.4238547 -16.8019848 49  150 8.2682993 216.1829079 50  151 6.4014518 152.6354187 51  152 5.3088876 115.4445340 52  153 1.6174395 -10.2123597 53  154 5.9485336 137.2180844 54  155 3.1546825 42.1153913 55  156 5.3005320 115.1601084 56  157 2.4789963 19.1150348 57  158 0.9767535 -32.0213123 58  159 -1.9511170 -131.6860236 59  160 6.9170995 170.1880667 60  161 5.2855369 114.6496771 61  162 4.4087823 84.8049493 62  163 3.1037112 40.3803307 63  164 1.5224966 -13.4442152 64  165 4.4883154 87.5122549 65  166 2.4841338 19.2899154 66  167 7.4551766 188.5042113 67  168 5.0847232 107.8139774 68  169 11.1806196 315.3182908 69  170 2.0649303 5.0202263 70  171 6.3135271 149.6424628 71  172 2.1035187 6.3337763 72  173 1.0137172 -30.7630659 73  174 4.3228148 81.8786157 74  175 7.6344148 194.6054803 75  176 4.3454526 82.6492051 76  177 6.6418251 160.8177274 77  178 1.4828424 -14.7940461 78  179 8.4935757 223.8513183 79  180 13.3411820 388.8638366 80  181 6.9229680 170.3878306 81  182 4.1925156 77.4432303 82  183 -0.3873840 -78.4565514 83  184 1.0014070 -31.1821045 84  185 4.8278364 99.0695505 85  186 6.5873198 158.9623664 86  187 11.1557332 314.4711578 87  188 2.3726048 15.4934678 88  189 0.5684373 -45.9203951 89  190 8.1500537 212.1578262 90  191 6.4236014 153.3893900 91  192 5.3376159 116.4224441 92  193 6.5534456 157.8092899 93  194 5.2696975 114.1105023 94  195 1.4119448 -17.2073978 95  196 4.5616554 90.0087510 96  197 10.9431095 307.2334460 97  198 6.3739809 151.7003083 98  199 4.3079963 81.3741929 99  200 4.2953062 80.9422246 100 |
|  |
| |  | | --- | | > | |

We will draw lines for each group that is made.

Like, group 1 – draw a line, group 2 – draw a line, etc.

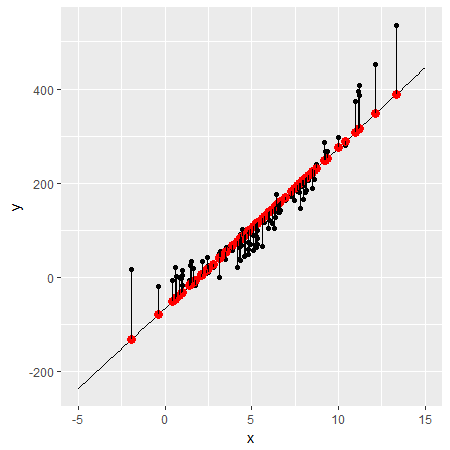
ggplot()+

geom\_point(data=dat, aes(x=x, y=y))+

stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=f)+

geom\_point(data=means, aes(x=x, y=y), color='red', size=3)+

geom\_line(data = groups, aes(x=x, y=y, group=group))



As you can see the lines are distances and those are called as residuals. If we want to get a measure of how good this line fits then we can square and add them up. That is called as residuals sum of squares. Sounds cool, let’s do that.

We will subtract the y values, that is how we will get distance.

> sum((dat$y-means$y)^2)

If you remember, dat has all the y values which are either above or below

And groups dataframe has all the y values which are ON the line.

|  |
| --- |
| [1] 158423.5 |
| We get residual sum of squares as 158423.5. That is a measure of how well this line fits.  **Best fitting line for polynomial degree 2**  > lm(y~x+I(x^2), data = dat)  Call:  lm(formula = y ~ x + I(x^2), data = dat)  Coefficients:  (Intercept) x I(x^2)  -0.5685 0.9719 2.9522  The coefficient for x^2 is 2.9522  The coefficient for x is 0.9719  My new function is  f <- function(x){  return(2.9522\*x^2 + 0.9719-0.5685)  }   |  | | --- | | > ggplot()+  + geom\_point(data=dat, aes(x=x, y=y))+  + stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=f) | |  | | |  | | --- | | > | | |
| |  | | --- | |  | |
| > head(means)  x y group  1 7.2940069 183.01800 1  2 7.7541205 198.68026 2  3 0.4429789 -50.19100 3  4 4.9272564 102.45381 4  5 6.4762602 155.18190 5  6 4.2627953 79.83555 6 |
|  |
| |  | | --- | | > | |

We have to change the y co-ordinates here

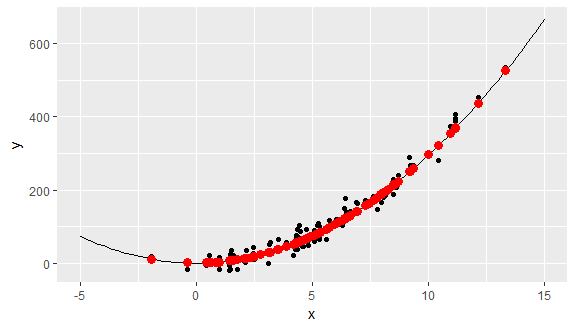
|  |
| --- |
| > means$y <- f(means$x) |
|  |
| |  | | --- | |  | |

ggplot()+

geom\_point(data=dat, aes(x=x, y=y))+

stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=f)+

geom\_point(data=means, aes(x=x, y=y), color='red', size=3)



Let’s draw residuals as well.

|  |
| --- |
| > groups <- rbind(dat,means) |
|  |
| |  | | --- | |  | |

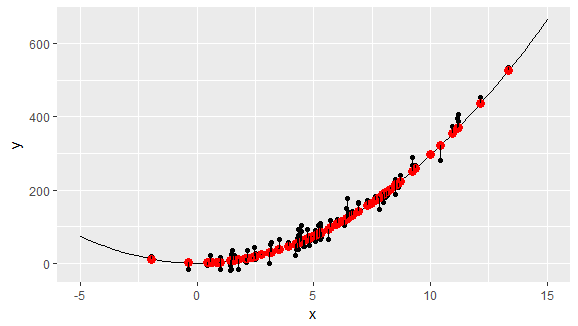
ggplot()+

geom\_point(data=dat, aes(x=x, y=y))+

stat\_function(data=data.frame(x=c(-5, 15)), aes(x=x), fun=f)+

geom\_point(data=means, aes(x=x, y=y), color='red', size=3)+

geom\_line(data = groups, aes(x=x, y=y, group=group))



We have visualization of residuals here.

Now we will also calculate residuals sum of squares.

|  |
| --- |
| > sum((dat$y - means$y)^2)  [1] 37138.18 |
|  |
| |  | | --- | | > | |